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: "Сучасні методи шифрування збережених даних в ОС Linux"

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Панасюк Максим Валентинович
підпис (прізвище та ініціали)

Керівник Стадник М. А.
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д. І.
підпис (прізвище та ініціали)

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

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2024

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)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Панасюку Максиму Валентиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Сучасні методи шифрування збережених даних в ОС Linux

Керівник роботи Стадник Марія Андріївна, к.т.н., доцент кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «15» 04 2024 року № 4/7-350

2. Термін подання студентом завершеної роботи 12.06.2024

3. Вихідні дані до роботи Вимоги до безпеки даних в операційній системі Linux. Операційна система Oracle Linux.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Огляд криптографічних функцій та методів
2. Методи шифрування збережених даних в операційній системі Oracle Linux
3. Тестування методів шифрування в операційній системі Linux
4. Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, задачі. Симетричне шифрування. Режими роботи симетричного блокового шифрування. Потокове шифрування. Асиметричне шифрування. Хеш-функції. Генератор псевдовипадкових чисел. Операційна система Oracle Linux. Алгоритми шифрування та використання LUKS. Використання EncFS. Алгоритми шифрування в EncFS. Процедура шифрування даних за допомогою LUKS. Процедура шифрування даних за допомогою EncFS. Зміна параметрів шифрування. Сумісність шифрування.
Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С. Ю., к.т.н. доцент кафедри МТ		

7. Дата видачі завдання 29.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.01.2024	
2.	Опрацювання джерел в галузі дослідження	02.02 – 30.01	
3.	Оформлення розділу «Огляд криптографічних функцій та методів»	21.02 – 10.03	
4.	Оформлення розділу «Методи шифрування збережених даних в операційній системі Oracle Linux»	11.03 – 25.03	
5.	Оформлення розділу «Тестування методів шифрування в операційній системі Oracle Linux»	10.04 – 05.05	
6.	Оформлення розділу «Безпека життєдіяльності, основи охорони праці»	10.05 – 21.05	
7.	Оформлення кваліфікаційної роботи	23.05 – 06.06	
8.	Нормоконтроль	06.06 – 10.06	
9.	Перевірка на плагіат	11.06 – 12.06	
10.	Попередній захист кваліфікаційної роботи	14.06 – 15.06	
11.	Захист кваліфікаційної роботи	27.06.2024	

Студент _____ Панасюк М. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Стадник М. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сучасні методи шифрування збережених даних в ОС Linux. // Кваліфікаційна робота ОР «Бакалавр» // Панасюк Максим Валентинович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2024 // С. 74, рис. – 39, табл. – 0, кресл. – 13, додат. – 2.

Ключові слова: Linux, LUKS, EncFS, AES, Serpent, Twofish, Blowfish, Camellia, XTS, CBC, dm-crypt, cryptsetup, encfs.

У кваліфікаційній роботі бакалавра було досліджено методи шифрування збережених даних у операційній системі Oracle Linux. У першому розділі надано огляд криптографічних методів, включаючи симетричне та асиметричне шифрування, а також хеш-функцій та генератора псевдовипадкових чисел. У другому розділі детально розглянуто методи шифрування в операційній системі Oracle Linux, зокрема LUKS та EncFS, а також їх алгоритми та параметри. Третій розділ присвячено практичній реалізації шифрування даних з використанням LUKS та EncFS на прикладі USB-накопичувача. Продемонстровано процес зміни алгоритмів шифрування та їх параметрів, а також процес розшифрування EncFS в операційній системі Windows. Результати дослідження підтверджують ефективність та надійність використаних методів шифрування даних у стані спокою в операційній системі Oracle Linux.

ANNOTATION

Modern methods of encryption of stored data in the Linux OS. // Thesis of educational level "Bachelor"// Maksym Panasiuk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБс-42 // Ternopil, 2024 // P. 74, fig. - 39, tab. - 0, chair. - 13, added. - 2.

Keywords: Linux, LUKS, EncFS, AES, Serpent, Twofish, Blowfish, Camellia, XTS, CBC, dm-crypt, cryptsetup, encfs.

In the bachelor's thesis, the methods of encryption of stored data in the Oracle Linux operating system were investigated. The first chapter provides an overview of cryptographic techniques, including symmetric and asymmetric encryption, as well as hash functions and pseudorandom number generators. In the second chapter, encryption methods in the Oracle Linux operating system, including LUKS and EncFS, as well as their algorithms and parameters, are discussed in detail. The third section is devoted to the practical implementation of data encryption using LUKS and EncFS on the example of a USB drive. The process of changing encryption algorithms and their parameters, as well as the process of EncFS decryption in the Windows operating system, is demonstrated. The results of the study confirm the effectiveness and reliability of the used data encryption methods at rest in the Oracle Linux operating system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ECB — Electronic Codebook

CBC — Cipher Block Chaining

CFB — Cipher Feedback

OFB — Output Feedback

CTR — Counter Mode

GCM — Galois/Counter Mode

XTS — XEX-Based Tweaked Codebook Mode With Ciphertext Stealing

MAC — Message Authentication Code

AE — Authenticated Encryption

AD — Associated Data

AEAD — Authenticated Encryption with Associated Data

TLS — Transport Layer Security

IPsec — Internet Protocol Security

GF — Galois Field

GMAC — Galois message authentication code

TE — Tweakable Encryption

XEX — XOR - encrypt - XOR

AES — Advanced Encryption Standard

nonce — number-only-used-once

RSA — Rivest-Shamir-Adleman

ECC — Elliptic Curve Cryptography

ECDSA — Elliptic Curve Digital Signature Algorithm

DH — Diffie-Hellman

ECDH — Elliptic Curve Diffie-Hellman

STS — Station-to-Station

DSA — Digital Signature Algorithm

MD5 — Message Digest Algorithm 5

SHA-1	—	Secure Hash Algorithm 1
SHA-2	—	Secure Hash Algorithm 2
SHA-3	—	Secure Hash Algorithm 3
SHA-3	—	Secure Hash Algorithm 3
PRNG	—	Pseudorandom Number Generator
UEK	—	Unbreakable Enterprise Kernel
RPM	—	Red Hat Package Manager
LUKS	—	Linux Unified Key Setup
SPN	—	Substitution-Permutation Network
PBKDF2	—	Password-Based Key Derivation Function 2
EncFS	—	Encrypted File System
EPEL	—	Extra Packages for Enterprise Linux
FUSE	—	Filesystem in Userspace

ВСТУП

Криптографія є ключовим елементом забезпечення безпеки у віртуальному та фізичному світі. Її застосування охоплює широкий спектр сфер, починаючи від забезпечення конфіденційності особистих повідомлень та закінчуючи захистом критично важливої інформації в банківській системі та урядових установах. Операційна система Linux використовується в багатьох галузях, включаючи корпоративні системи та інфраструктуру хмарних обчислень. Захист конфіденційної інформації в цій операційній системі є критично важливим моментом. Недостатня безпека може призвести до серйозних наслідків для організацій.

Метою даного дослідження є аналіз та практична реалізації методів шифрування даних в операційній системі Oracle Linux. Зокрема симетричного та асиметричного шифрування, а також методів забезпечення цілісності даних.

Основними задачами дослідження є:

- огляд криптографічних функцій та методів;
- огляд доступних вбудованих засобів шифрування збережених даних в Oracle Linux та вивчення їхніх можливостей;
- детальний аналіз процесів шифрування та розшифрування;
- огляд та дослідження можливостей вбудованих методів шифрування, таких як LUKS та EncFS;
- практична реалізація методів шифрування LUKS та EncFS;
- оцінка ефективності використання вбудованих засобів шифрування в операційній системі Linux.

Об'єктом дослідження є методи шифрування даних в операційній системі Oracle Linux, включаючи LUKS та EncFS, а також їх практичне застосування.

Предметом дослідження є процеси шифрування та розшифрування даних в Oracle Linux, включаючи аналіз алгоритмів, параметрів та інструментів, які використовуються для забезпечення безпеки даних.

Результати дослідження можуть бути корисними для адміністраторів систем, які відповідають за безпеку даних в операційній системі Linux.

Розуміння різних методів шифрування та їх практичні застосування дозволить їм краще захистити конфіденційну інформацію в системі, зменшуючи ризик несанкціонованого доступу.

РОЗДІЛ 1 ОГЛЯД КРИПТОГРАФІЧНИХ ФУНКЦІЙ ТА МЕТОДІВ

Криптографія - це наука про забезпечення конфіденційності, цілісності та автентичності інформації шляхом застосування різних математичних методів та алгоритмів. Основною метою криптографії є забезпечення безпеки в обміні даними та зберіганні інформації шляхом шифрування даних таким чином, щоб тільки вповноважені особи могли їх розшифрувати [1].

Шифрування даних - це процес перетворення звичайного тексту (відкритий текст) в зашифрований текст за допомогою спеціальних алгоритмів та ключів. Його основна мета - забезпечити конфіденційність інформації, унеможлививши несанкціонований доступ до неї. Основні принципи шифрування полягають у використанні криптографічних алгоритмів для перетворення відкритого тексту у шифрований текст і обернене перетворення для отримання відкритого тексту з зашифрованого. Алгоритми відіграють важливу роль у цьому процесі, оскільки вони визначають конкретний спосіб шифрування та дешифрування даних.

Застосування шифрування даних дозволяє зберегти конфіденційність інформації навіть у випадку втрати або крадіжки пристроїв зберігання даних [2]. Це є невід'ємною складовою будь-якої комплексної системи безпеки даних і сприяє підвищенню рівня довіри та захищеності в інформаційному середовищі.

Хешування - це процес перетворення будь-якого об'єкта даних у фіксований рядок фіксованої довжини за допомогою хеш-функції. Основна властивість хеш-функції полягає в тому, що для однакового вхідного значення завжди буде отримуватися однаковий хеш-вихід, але будь-яка незначна зміна вхідних даних призведе до зміни хешу. Хеш-функції широко використовуються у криптографії для забезпечення цілісності даних, створення цифрових підписів, зберігання паролів користувачів.

1.1 Симетричне шифрування

Симетричне шифрування - це метод шифрування, де той самий ключ використовується як для шифрування, так і для розшифрування даних [3] (див. рисунок 1.1).

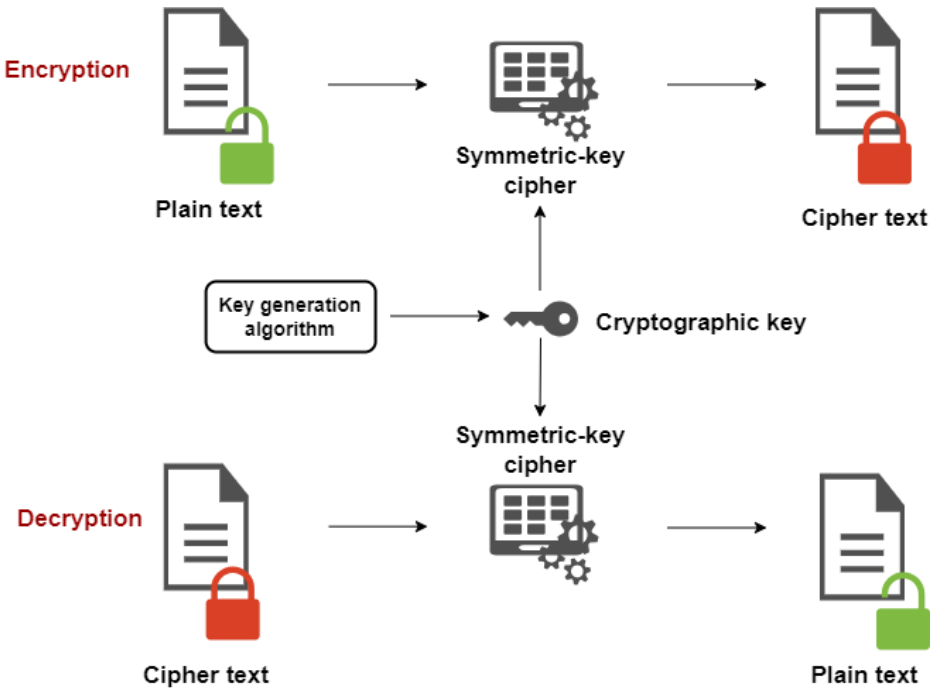

Рисунок 1.1 – Симетричне шифрування

Основні переваги симетричного шифрування включають його високу швидкодію та ефективність. Це робить його особливо корисним для шифрування великих обсягів даних. Однак головний недолік полягає в розповсюдженні ключа. Якщо ключ потрапляє у руки зловмисника, всі дані є під загрозою несанкціонованого доступу.

1.1.1 Блокове шифрування

Блокове шифрування - це метод шифрування, в якому вхідні дані розбиваються на фіксовані блоки однакового розміру, які потім шифруються окремо. Кожен блок даних обробляється алгоритмом шифрування, який використовується з одним і тим же ключем для кожного блоку. Такий підхід дозволяє шифрувати та розшифровувати великі обсяги даних швидко та

ефективно [5]. Блоки даних мають фіксований розмір, який зазвичай вимірюється у бітах або байтах. Зазвичай використовуються блоки розміром 64 або 128 біт. Ключ шифрування - це секретний параметр, який використовується для шифрування та розшифрування даних. Ключ визначає складність шифрувального алгоритму [6].

Є дві основні причини, чому блоковий шифр сама по собі не може бути практичною безпечною схемою шифрування. Перший полягає в тому, що ми хотіли б зашифрувати будь-які довгі повідомлення, але блоковий шифр приймає лише вхідні дані фіксованої довжини. Інша полягає в тому, що якщо дані повторюються, зашифрований текст є однаковим (тобто він є детермінованим). Щоб вирішити ці проблеми, алгоритм шифрування має бути рандомізованим.

Існує кілька стандартних способів (режимів роботи) побудови алгоритму шифрування за допомогою блокового шифру: ECB, CBC, CFB, OFB, CTR, GCM, XTS та інші [7]. Кожен режим має свої особливості щодо застосування ключа та вектора ініціалізації IV.

Режим ECB є найпростішим із усіх. На рисунку 1.2 представлена схема режиму роботи ECB.

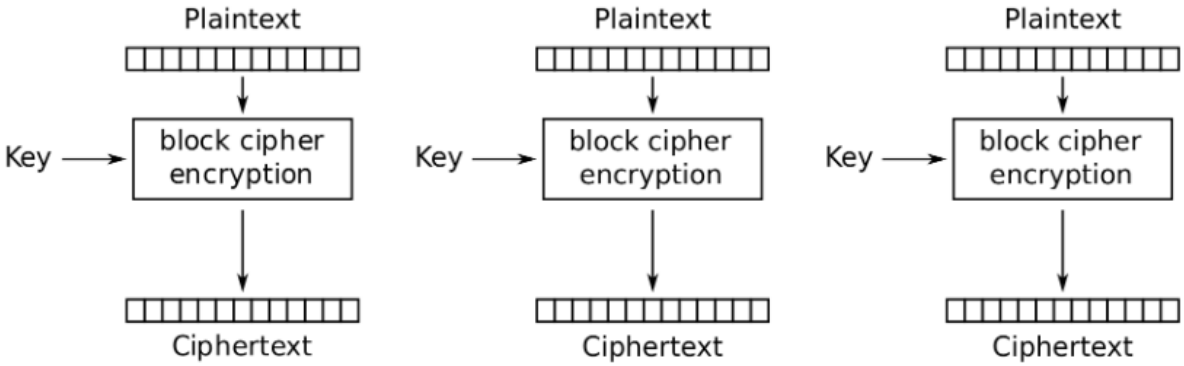

Рисунок 1.2 – Схема режиму роботи ECB

Відкритий текст поділений на блоки відповідно до довжини блоку. Кожен блок буде зашифровано тим самим ключем і тим самим алгоритмом. Отже, якщо ми зашифруємо ідентичний відкритий текст, ми отримаємо ідентичний зашифрований текст. Саме тому цей режим вважається ненадійним з міркувань

безпеки. Шифрування та дешифрування є незалежним ми можемо шифрувати дешифрувати дані паралельно.

На рисунку 1.3 представлена схема режиму роботи CBC.

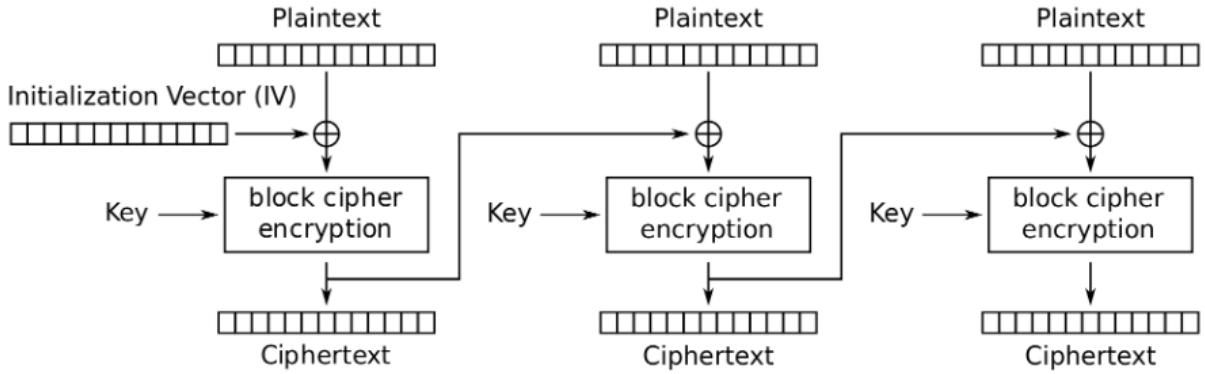

Рисунок 1.3 – Схема режиму роботи CBC

CBC працює подібно до ECB, але прив’язує кожен блок до попереднього. Для першого блоку використовується вектор ініціалізації (IV). IV має такий самий розмір, як і блок. В подальшому кожен блок відкритого тексту об’єднується XOR з попереднім блоком зашифрованого тексту перед шифруванням. Навіть якщо шифрування буде використано двічі для того самого відкритого тексту та ключа, зашифрований текст кожного разу буде іншим через випадковий IV та ланцюги попередніх зашифрованих блоків. Це гарантує, що ідентичні блоки відкритого тексту не будуть зашифровані в ідентичні блоки зашифрованого тексту.

На рисунку 1.4 представлена схема режиму роботи CFB.

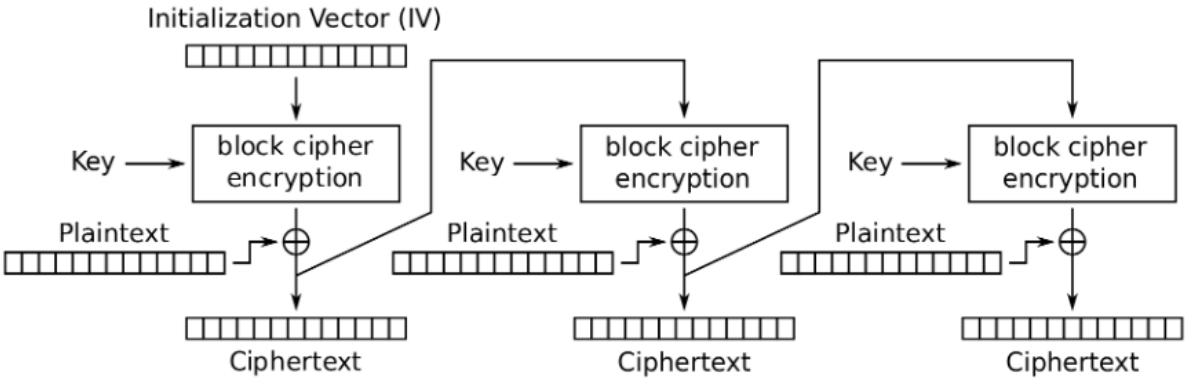

Рисунок 1.4 – Схема режиму роботи CFB

Спочатку CFB зашифрує IV, потім виконує XOR із блоком відкритого тексту, щоб отримати зашифрований текст. Цей режим не шифрує відкритий текст напряму, він просто використовує зашифрований текст для XOR із відкритим текстом, щоб отримати зашифрований текст. Цей режим схожий на CBC, тому якщо є помилка в блоці, це вплине на всі наступні блоки.

На рисунку 1.5 представлена схема режиму роботи OFB.

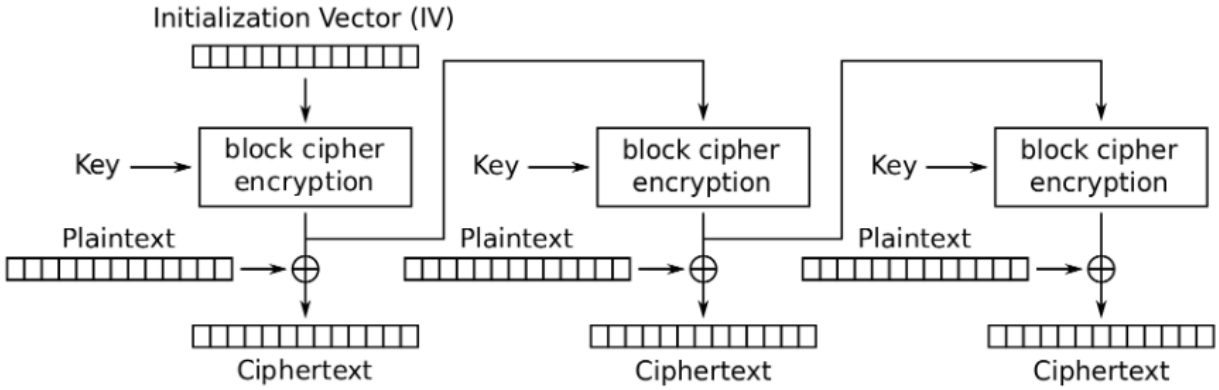

Рисунок 1.5 – Схема режиму роботи OFB

У цьому режимі шифрується IV перший раз. Потім виконується XOR із блоком відкритого тексту, щоб отримати зашифрований текст. Помилка в блоці не вплине на всі наступні блоки.

На рисунку 1.6 представлена схема режиму роботи CTR.

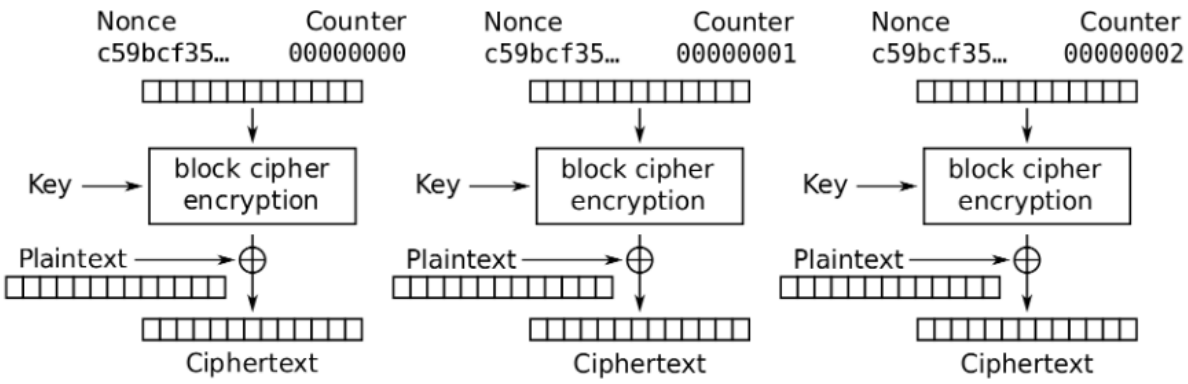

Рисунок 1.6 – Схема режиму роботи CTR

Режим CTR практично перетворює блоковий шифр на потоковий. Кожен блок повідомлення не повинен мати однакове значення лічильника (counter). У режимі CTR IV іноді перейменовується на nonce. Число nonce використовується лише один раз та є унікальним. Nonce та counter шифруються блоковим шифрам для формування потоку ключів. Потім відкритий текст та потік ключів по одному біту за допомогою XOR перетворюється на зашифрований текст. Усі блоки шифрування використовують однаковий ключ шифрування. Помилка в блоці не вплине на всі наступні блоки.

В режимах CFB, CTR і OFB алгоритм дешифрування використовує функцію шифрування блокового шифру замість функції дешифрування. Відкритий текст ніколи не проходить через шифрування блоковим шифром, тому дешифрування блоковим шифром ніколи не використовується.

На рисунку 1.7 представлена схема режиму роботи GCM.

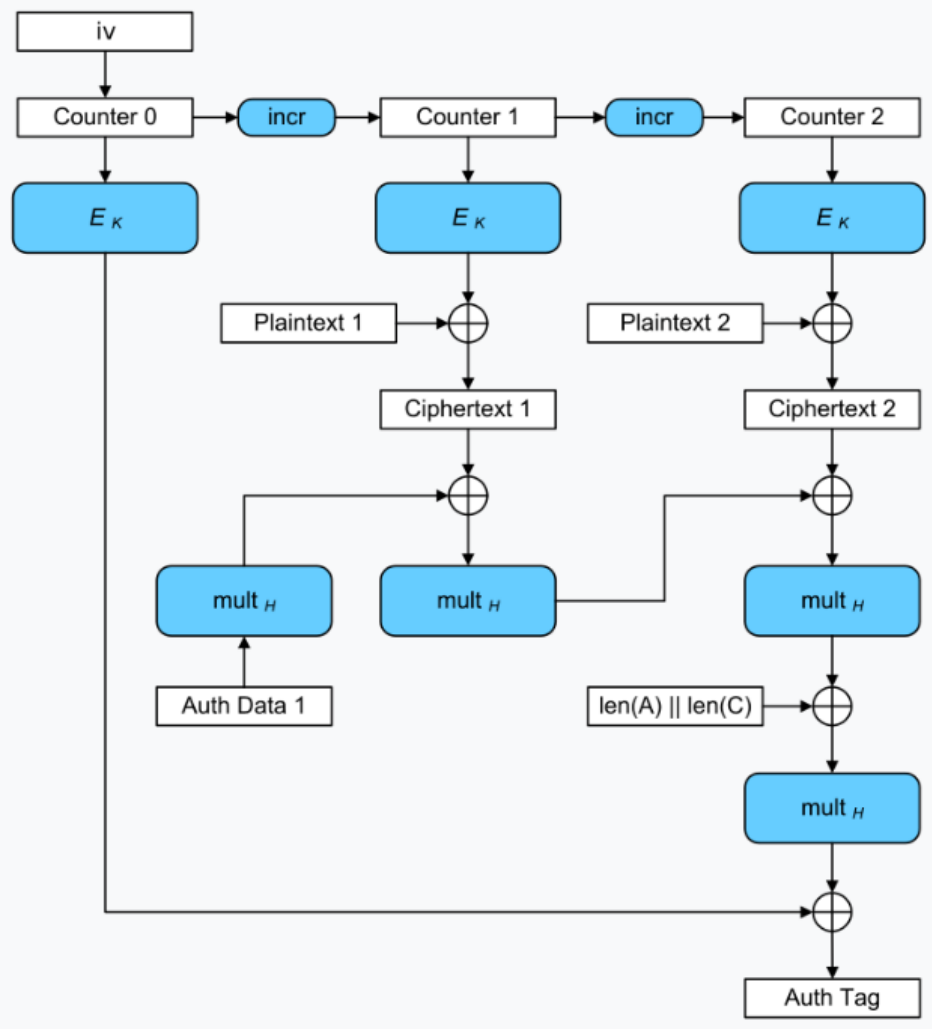

Рисунок 1.7 – Схема режиму роботи GCM

Режим GCM по суті базується на CTR, але включає функцію коду автентифікації повідомлення (MAC). Це робиться для того, щоб під час дешифрування можна було перевірити, чи дані були змінені (автентичність і цілісність даних). Це називається автентифіковане шифрування (AE).

Автентичність незашифрованих даних, таких як дані заголовка (AD), також перевіряється за допомогою функції MAC GCM. Це називається автентифіковане шифрування з пов'язаними даними (AEAD). AEAD - це форма криптографічного захисту даних, яка комбінує в собі процеси шифрування і автентифікації. AEAD дозволяє не тільки зашифрувати дані, а й забезпечити їх автентичність та цілісність. Крім того, AEAD дозволяє включати в шифртекст додаткові дані, які не шифруються, але автентифікуються разом з ними.

Для перевірки того, чи змінилися дані, використовується метод хешування GHASH. Метод хешування GHASH - це частина автентифікаційного режиму шифрування GCM, який використовується для захисту даних у протоколах забезпечення безпеки, таких як TLS і IPsec. Автентифікаційний код GHASH обчислюється за допомогою поля Галуа (GF), що є поліноміальним кільцем над скінченим полем, зазвичай GF (2^{128}), і ключа автентифікації (який зазвичай співпадає з ключем шифрування).

GHASH використовується для обчислення аутентифікаційного тегу, або хешу, для захисту даних в режимі GCM. Цей тег використовується для перевірки цілісності даних. Обчислення GHASH засноване на математичних операціях у полі Галуа, що дозволяє виконувати операції множення і додавання з дуже великими числами за допомогою операцій XOR і зсуву бітів. GHASH використовується в широкому спектрі застосувань, де важливо забезпечити як конфіденційність, так і цілісність даних, включаючи безпеку мережевого трафіку, збереження даних та інші області, де захист даних є критичним.

Після шифрування метод MAC генерує тег автентифікації (T), який також можна обчислити під час дешифрування. Якщо вони збігаються, можна вважати, що дані не змінено. GCM також можна використовувати для автентифікації

незашифрованих даних; це відоме як код автентифікації повідомлення Галуа (GMAC). GMAC є спеціалізованою формою GCM, призначеною для автентифікації даних, що не зашифровані. Це варіант використання GCM, де забезпечується лише аутентифікація без шифрування.

У GMAC використовується той же механізм, що й у GCM, але без процесу шифрування. Використовується ключ та блок даних для генерації тегу автентифікації, що забезпечує перевірку цілісності даних без необхідності їх шифрування. Це робить GMAC корисним у сценаріях, де необхідна автентифікація, але шифрування не потрібне або небажане, наприклад, для захисту заголовків даних у мережевих протоколах.

На рисунку 1.8 представлена схема режиму роботи XTS.

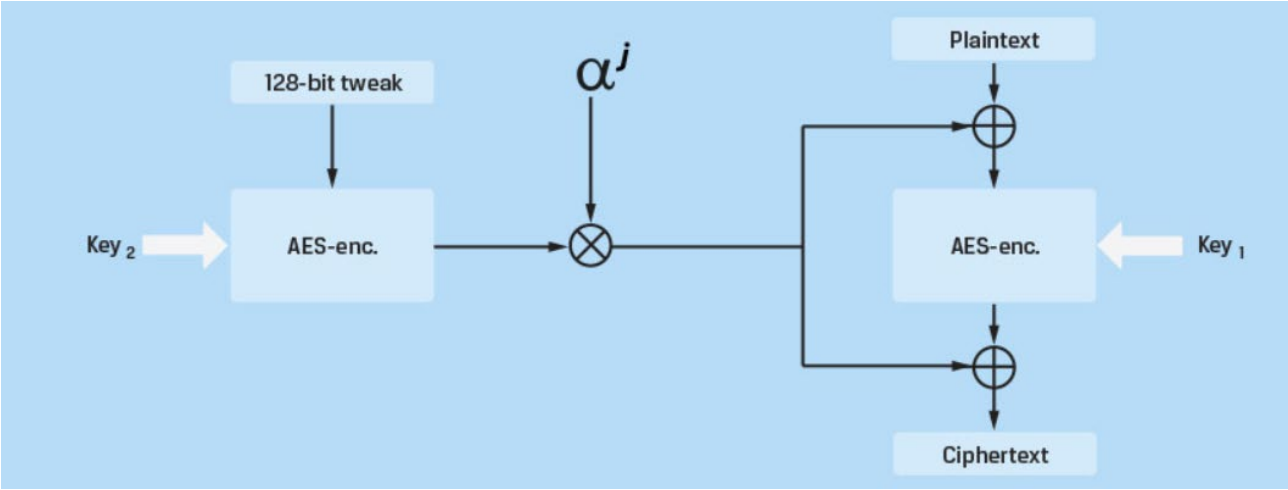

Рисунок 1.8 – Схема режиму роботи XTS

XTS - це режим, який призначений для шифрування даних на диску. Велике значення має поділ дисків на сектори, оскільки кожен сектор має бути зашифрований і дешифрований надійно, окремо та з високою продуктивністю. Налаштоване шифрування (TE) використовується в шифруванні диска, щоб включити сектор у процес шифрування. Це дозволяє шифрувати весь диск за допомогою того самого ключа та комбінації адреси сектора та індексу блоку всередині сектора, яка використовується як додатковий параметр або ключ (tweak). Tweak забезпечує додатковий рівень безпеки та допомагає гарантувати,

що шифрування двох однакових блоків даних з різними адресами дасть різні результати.

Основним режимом для XTS є XEX, який використовує два ключі для шифрування та tweak.

Key 2 - це ключ, який використовується разом із алгоритмом шифрування для перетворення tweak в унікальний код. Це значення додається до кожного блоку даних перед їхнім шифруванням для внесення унікальності.

Операції XOR використовуються для комбінації різних вхідних даних, таких як α^j з plaintext та ciphertext з попереднім результатом шифрування.

Key 1 - це ключ, який використовується для шифрування комбінованого вхідного сигналу (plaintext, після XOR з α^j). Результат - це зашифрований текст, який потім комбінується знову з α^j для отримання кінцевого зашифрованого блоку (ciphertext).

В режимі XTS α^j відіграє роль множника поля Галуа, який є частиною механізму, що додає додатковий рівень безпеки до кожного блоку даних. Tweak, який шифрується окремим ключем (Key 2) і служить для генерації множника поля Галуа α^j є вхідним параметром, який модифікується та використовується для створення множника.

Поле Галуа α представляє елемент скінченного поля (зазвичай $GF(2^{128})$ для XTS), який використовується для генерації серії множників, залежних від номера блоку j. Цей елемент поля є базовим елементом, що використовується для обчислень у полі. Індекс j вказує на порядковий номер блоку в рамках шифрування. В режимі XTS α^j використовується для модифікації кожного блоку перед його шифруванням та після шифрування. Це допомагає забезпечити додаткову випадковість і робить шифровані дані менш вразливими до певних типів атак.

Основні блокові шифри: AES, Blowfish, DES, Triple DES (3DES), Serpent, Camellia, Twofish.

1.1.2 Потокове шифрування

Потокове шифрування - це тип криптографічного шифрування, де кожен біт відкритого тексту шифрується окремо за допомогою ключа, що генерує послідовність псевдовипадкових бітів, відомих як потік шифру. Цей потік бітів потім комбінується з відкритим текстом за допомогою операції XOR щоб отримати шифрований текст [7].

На рисунку 1.9 представлена схема роботи потокового симетричного шифрування.

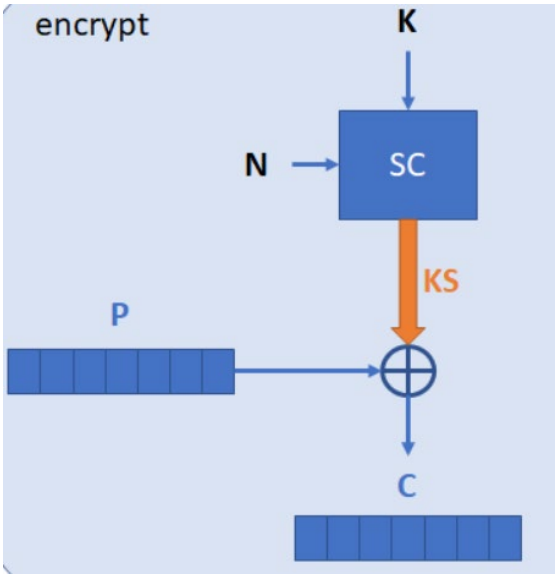

Рисунок 1.9 – Схема роботи потокового симетричного шифрування

Потоковий шифр працює методом шифрування кожного біта відкритого тексту P одним бітом так званого потоку ключів KS для створення зашифрованого тексту C. Потік ключів генерується з секретного ключа K та nonce N з використанням потокових шифрів. Ключовою частиною цього процесу є генерація потоку псевдовипадкових бітів на основі ключа шифрування K та початкового числа, відомого як унікальне випадково генероване число nonce.

Основні потокові шифри: RC4, Fortuna, Grain, Salsa20, A5/1, HC-128, ISAAC, ChaCha20 та Grain-128.

1.2 Асиметричне шифрування

Асиметричне шифрування, також відоме як криптографія з відкритим ключем, є методом шифрування, який використовує два різні, але математично пов'язані, ключі: відкритий ключ і приватний ключ [3]. Цей метод вирішує основні проблеми симетричного шифрування, такі як безпечна передача ключа між відправником і отримувачем (див. рисунок 1.10).

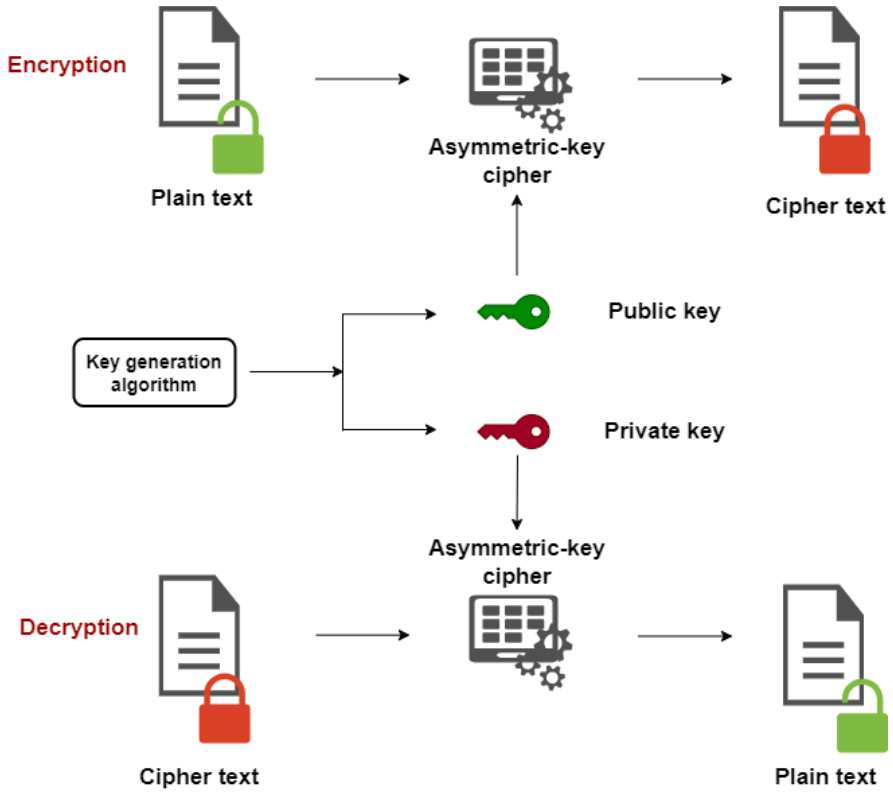

Рисунок 1.10 – Асиметричне шифрування

Відкритий ключ (Public Key) - це ключ, який доступний всім і використовується для шифрування повідомлень. Він може бути вільно поширений і не вимагає захисту. Приватний ключ (Private Key) - це ключ, який тримається в таємниці і використовується для розшифрування повідомлень, зашифрованих за допомогою відповідного відкритого ключа. Він має бути захищений і доступний лише власнику пари ключів.

Відправник використовує відкритий ключ отримувача для шифрування повідомлення. Лише приватний ключ отримувача може розшифрувати це повідомлення, забезпечуючи конфіденційність. Отримувач використовує свій приватний ключ для розшифрування повідомлення, зашифрованого відкритим

ключем. Це гарантує, що лише власник приватного ключа може отримати доступ до зашифрованої інформації.

Відправник може використовувати свій приватний ключ для підпису повідомлення. Отримувач може перевірити підпис, використовуючи відкритий ключ відправника. Це забезпечує автентичність та цілісність повідомлення.

Оскільки відкритий ключ може бути поширений безпечно, немає необхідності у безпечному каналі для обміну ключами, як у симетричному шифруванні. Приватний ключ тримається в таємниці, і навіть якщо відкритий ключ буде перехоплений, зашифровані повідомлення залишаться захищеними. Є можливість перевірки автентичності та цілісності повідомлень, що є важливим для запобігання підробці та несанкціонованим змінам.

Асиметричне шифрування зазвичай повільніше за симетричне, що робить його менш придатним для шифрування великих об'ємів даних. Тому часто використовують гібридні системи, де асиметричне шифрування застосовується для обміну симетричними ключами, а самі дані шифруються симетричними алгоритмами. Реалізація та управління ключами в асиметричному шифруванні є складнішими порівняно з симетричним шифруванням.

Основні потокові шифри: RSA, ECC.

RSA – це один з перших і найбільш використовуваних алгоритмів асиметричного шифрування. RSA базується на складності факторизації великих цілих чисел. Використовує пари ключів, що складаються з відкритого та приватного ключів. Відкритий ключ використовується для шифрування, а приватний – для розшифрування. RSA також використовується для створення цифрових підписів. Для сучасного рівня безпеки рекомендується використовувати ключі довжиною не менше 2048 біт.

ECC - базується на математичних властивостях еліптичних кривих. ECC забезпечує таку ж безпеку як RSA, але з меншими розмірами ключів. Наприклад, ключ ECC довжиною 256 біт еквівалентний за безпекою RSA-ключу довжиною 3072 біт. Що робить його більш ефективним для використання в мобільних пристроях та інших середовищах з обмеженими ресурсами. Для підпису та верифікації повідомлень за допомогою ECC використовується алгоритм ECDSA.

DH – це протокол обміну ключами, який дозволяє двом сторонам створити спільний секретний ключ, навіть якщо вони спілкуються через незахищений канал. Цей секретний ключ може бути використаний для симетричного шифрування подальших комунікацій. DH на еліптичних кривих (ECDH) передбачає використання еліптичних кривих для протоколу DH, що значно підвищує безпеку при меншому розмірі ключа. ECDH є варіантом DH, що базується на складності обчислення дискретного логарифма в групах точок еліптичних кривих. STS є розширенням DH, яке включає автентифікацію за допомогою цифрових підписів, щоб запобігти атакам типу людина посередині.

DSA – це стандарт для цифрових підписів, розроблений NIST. DSA використовується виключно для створення цифрових підписів і не підходить для шифрування даних. Використовує пару ключів для підпису і верифікації документів. Підпис гарантує автентичність і цілісність повідомлення. DSA може бути повільнішим у порівнянні з деякими іншими алгоритмами цифрового підпису, такими як ECDSA.

Асиметричне шифрування є важливим інструментом в сучасній криптографії, забезпечуючи безпечний обмін ключами та автентифікацію. Використання асиметричних алгоритмів, таких як RSA, ECC і DH, дозволяє створювати надійні системи захисту інформації, що забезпечують конфіденційність, цілісність і автентичність даних.

1.5 Хеш-функції

Хеш-функції - це фундаментальний елемент сучасної криптографії. Вони використовуються для створення стислого, фіксованого розміру вихідного значення (хешу) з вхідних даних будь-якого розміру [8] (див. рисунок 1.11).

Рисунок 1.11 – Принцип роботи хеш-функції

Хеш-функції знаходять застосування в різних областях, включаючи зберігання паролів, перевірку цілісності даних, цифрові підписи та інші криптографічні протоколи [9].

Для одних і тих самих вхідних даних хеш-функція завжди повертає один і той самий хеш. Обчислення хешу має бути швидким і ефективним навіть для великих обсягів даних. Хеш-функція перетворює дані довільного розміру у вихід фіксованого розміру. Неможливо знайти два різних вхідних повідомлення, які б дали той самий хеш (стійкість до колізій). Неможливо (або дуже складно) відновити початкові дані з їх хешу (одностороння функція).

Основні алгоритми хеш-функцій: MD5, SHA-1, SHA-2, SHA-3.

MD5 генерує 128-бітний хеш. Вхідні дані обробляються блоками по 512 біт. SHA-1 генерує 160-бітний хеш. Вхідні дані обробляються блоками по 512 біт. SHA-2 генерує хеші різних розмірів: 224, 256, 384 і 512 біт. Вхідні дані обробляються блоками по 512 (SHA-256) або 1024 (SHA-512) біт. SHA-3 генерує хеші різних розмірів: 224, 256, 384 і 512 біт.

Хеш-функцій використовується для забезпечення цілісності даних при передачі або зберіганні. Хеш повідомлення порівнюється з очікуваним значенням, щоб перевірити, чи не були дані змінені. Хеш-функції використовуються для зберігання паролів у зашифрованому вигляді. Замість зберігання самих паролів зберігається їх хеш, що ускладнює компрометацію паролів у разі витоку даних. Хеш-функцій також використовується для створення унікальних ідентифікаторів документів та цифрових сертифікатів, що дозволяє перевіряти автентичність і цілісність документів.

Хеш-функції забезпечують ефективний і надійний спосіб перевірки цілісності даних, автентифікації користувачів і зберіганні конфіденційної інформації.

1.4 Генератор псевдовипадкових чисел

Криптографія часто вимагає випадковості. Наприклад, симетричні ключі, як правило, є випадково згенерованими бітами, а випадкові IV і nonces потрібні для побудови безпечних режимів з'єднання блокових шифрів [1]. Створення справжньої, неупередженої випадковості обчислювально дороге. Справжня випадковість зазвичай вимагає вибірки даних з непередбачуваного фізичного процесу, наприклад непередбачуваної схеми на ЦП, випадкових шумових сигналів або мікросекунди, через яку користувач натискає клавішу. Ці джерела можуть бути упередженими та передбачуваними, що ускладнює створення неупередженої випадковості.

Замість використання дорогої справжньої випадковості кожного разу, коли криптографічний алгоритм вимагає випадковості, ми натомість використовуємо псевдовипадковість. Псевдовипадкові числа генеруються детерміновано за допомогою алгоритму, але вони виглядають випадковими. Зокрема, хороший алгоритм псевдовипадкових чисел генерує біти, які обчислювально неможливо відрізнити від справжніх випадкових бітів – не існує ефективного алгоритму, який дозволив би зловмиснику відрізнити псевдовипадкові біти від справді випадкових бітів.

Генератор псевдовипадкових чисел (PRNG) - це алгоритм для генерації послідовності чисел, властивості яких наближені до властивостей послідовностей випадкових чисел. Послідовність, згенерована PRNG, визначається початковим значенням seed. PRNG зазвичай складається з функції ініціалізації (initialization function), стану (state), що є послідовністю бітів обмеженої довжини, функції переходу (function) та функції виведення (output function) (див.рисунок 1.12).

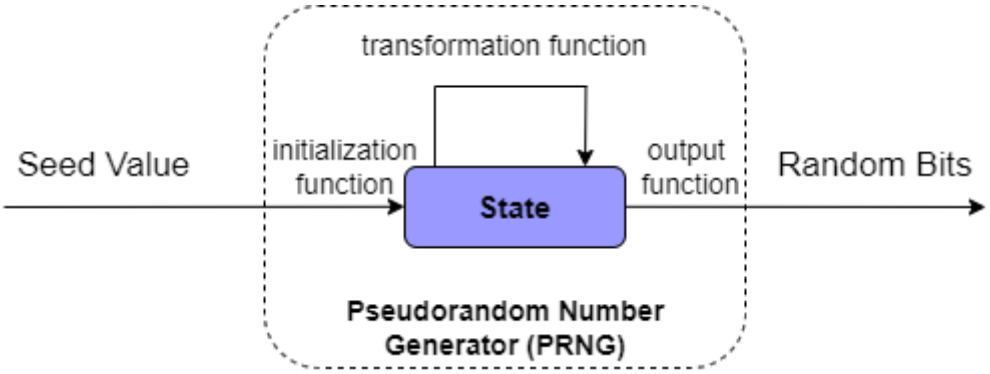

Рисунок 1.12 – Схема роботи генератор псевдовипадкових чисел

Псевдовипадковий генератор повинен генерувати послідовність, яка відповідає різним статистичним властивостям, щоб забезпечити безпеку та непередбачуваність. Кожне значення в послідовності повинно мати однакову ймовірність виникнення, тобто жодне значення не повинно виникати частіше або рідше за інші. Кожне значення в послідовності повинно бути незалежним від інших значень, тобто не повинно бути можливості передбачити наступне значення на основі попередніх. Послідовність повинна виглядати як випадкова інформація, навіть якщо вона фактично є детермінованою. Послідовність повинна бути стійкою до атак криптоаналізу, що означає, що навіть знання часткової послідовності не дозволить легко передбачити інші значення.

Враховуючи ці властивості, розробники PRNG створюють алгоритми, які забезпечують потрібний рівень випадковості та непередбачуваності, що дозволяє їм використовуватися в різних криптографічних застосунках та інших областях, де потрібна випадкова генерація даних.

1.5 Висновки до розділу

В першому розділі було проведено огляд криптографічних функцій та методів. Здійснено детальний огляд симетричного шифрування. Описано його переваги та недоліки. Описано режими роботи блокового шифру ECB, CBC, CFB, OFB, CTR, GCM та XTS. Здійснено порівняння блокового та потокового симетричного шифрування.

Здійснено детальний огляд асиметричного шифрування. Описано принцип використання відкритого та приватного ключів в асиметричному шифруванні та в цифровому підписі. Проведено огляд асиметричних шифрів RSA та ECC.

Описано принцип роботи та можливості застосування хеш-функцій та генератора псевдовипадкових чисел.

РОЗДІЛ 2 МЕТОДИ ШИФРУВАННЯ ЗБЕРЕЖЕНИХ ДАНИХ В ОПЕРАЦІЙНІЙ СИСТЕМІ ORACLE LINUX

2.1 Операційна система Oracle Linux

Oracle Linux – це дистрибутив операційної системи Linux, який розроблений і підтримується компанією Oracle. Ця операційна система орієнтована на корпоративні середовища, забезпечуючи стабільну, безпечну та продуктивну платформу для розгортання серверних додатків, баз даних та інших корпоративних сервісів [10].

Oracle Linux базується на вихідному коді Red Hat Enterprise Linux, що забезпечує високу сумісність із цим дистрибутивом. Це дозволяє користувачам без проблем використовувати ті ж самі пакети, налаштування та інструменти. Oracle Linux пропонує власне ядро Unbreakable Enterprise Kernel, яке оптимізоване для високої продуктивності та надійності в корпоративних середовищах. UEK включає оновлення та оптимізацію, спрямовані на покращення продуктивності баз даних Oracle та інших корпоративних додатків.

Oracle надає комерційну підтримку для Oracle Linux, включаючи регулярні оновлення безпеки та технічну допомогу. Це робить Oracle Linux надійним вибором для критично важливих середовищ. Oracle Linux підтримує сучасні технології контейнеризації та віртуалізації, такі як Docker, Kubernetes і KVM. Це забезпечує гнучкість і масштабованість для розгортання додатків у різних середовищах. Oracle Linux використовує RPM та DNF для управління пакетами, що забезпечує легкість установки, оновлення та видалення програмного забезпечення.

2.2 Шифрування дисків на рівні блоків

2.2.1 Використання LUKS

LUKS - це стандарт, який визначає специфікацію для шифрування диску на рівні блоків в операційній системі Linux [11]. Він дозволяє шифрувати весь диск

або окремі розділи, що дозволяє забезпечити конфіденційність даних. LUKS забезпечує прозоре шифрування даних, що означає, що користувач може працювати з даними, як звичайно, навіть під час їх шифрування та розшифрування. LUKS дозволяє керувати ключами шифрування, включаючи можливість додавання, видалення та зміни ключів без необхідності розшифровування всього диску. LUKS підтримує використання різних методів автентифікації, таких як паролі, ключі або комбінації цих методів. Існують різні утиліти, які дозволяють працювати з LUKS, включаючи можливість зміни паролів, додавання або видалення ключів, розшифрування даних тощо. LUKS є відкритим стандартом, що робить його доступним для перевірки та аудиту, а також для реалізації в різних дистрибутивах Linux та інших системах.

LUKS забезпечує загальне сховище ключів у виділеній області на диску з можливістю використання кількох парольних фраз для розблокування збереженого ключа. LUKS2 розширює цю концепцію для більш гнучких способів зберігання метаданих, надлишкової інформації для забезпечення відновлення у разі пошкодження в області метаданих та інтерфейсу для зберігання зовнішніх керованих метаданих для інтеграції з іншими інструментами. Реалізація LUKS2 призначена для використання з шифруванням диска dm-crypt на основі Linux.

Dm-crypt - це механізм шифрування на рівні ядра Linux, який дозволяє користувачам монтувати зашифровану файлову систему [12]. Монтування файлової системи - це процес, під час якого файлова система приєднується до каталогу (точка монтування), що робить її доступною для операційної системи. Після монтування всі файли у файловій системі доступні для програм, однак ці файли зашифровані під час зберігання на диску.

Device mapper - це інфраструктура в ядрі Linux, яка забезпечує загальний спосіб створення віртуальних рівнів блокових пристроїв. Криптографічна ціль відображення пристроїв забезпечує прозоре шифрування блокових пристроїв за допомогою API шифрування ядра.

На рисунку 2.1 зображено зв’язок між програмою, файловою системою та dm-crypt. Dm-crypt знаходиться між фізичним диском і файловою системою, і дані, записані з операційної системи на диск, шифруються.

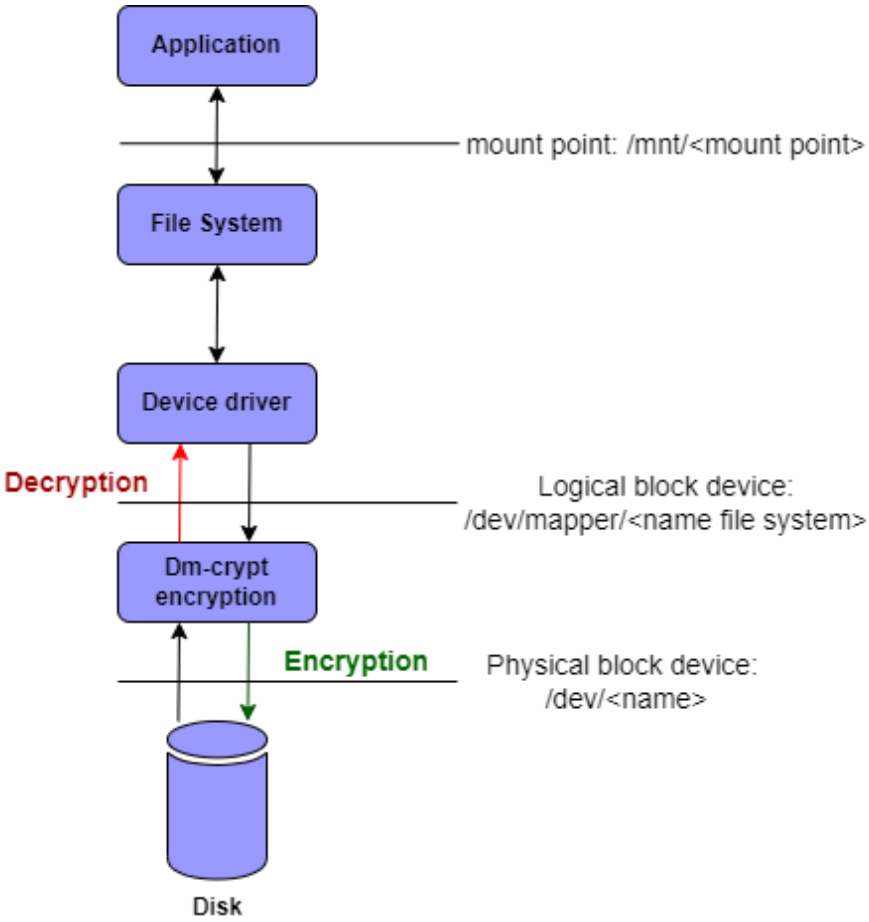

Рисунок 2.1 – Принцип взаємодії з dm-crypt

Програма не знає про шифрування на рівні диска. Програми використовують певну точку монтування для зберігання та отримання файлів. Ці файли шифруються під час зберігання на диску. Якщо диск втрачено або вкрадено, дані на диску недоступні.

На етапі встановлення проведемо шифрування диску операційної системи за допомогою LUKS. Шифрування диску під час встановлення Oracle Linux є доцільним кроком для забезпечення безпеки системи та захисту конфіденційних даних. На рисунку 2.2 зображено етап встановлення операційної системи з налаштуванням шифрування системного диску.

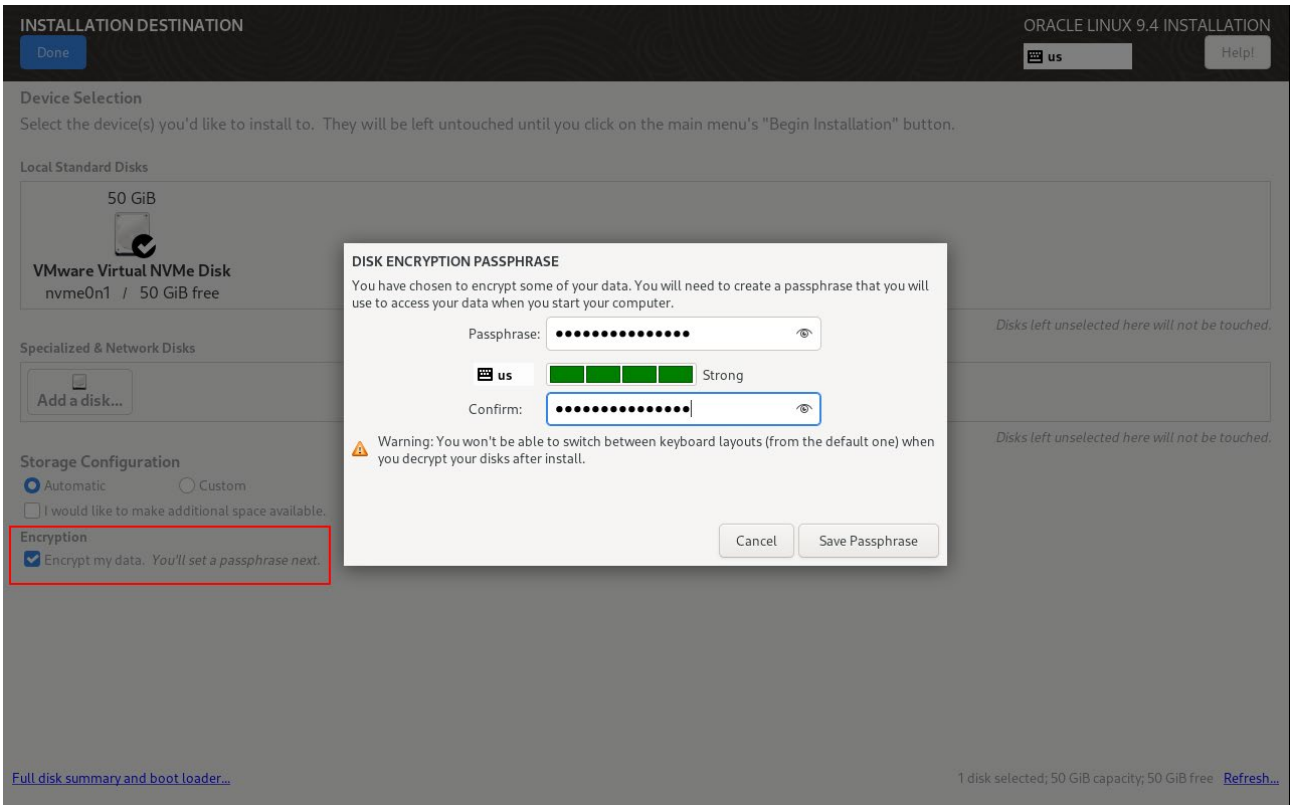

Рисунок 2.2 – Шифруванням системного диску на етапі встановлення Oracle Linux

Після встановлення операційної системи в менеджері дисків можна побачити, що системний розділ /dev/nvme0n1p2 зашифровано за допомогою LUKS (див. рисунок 2.3).

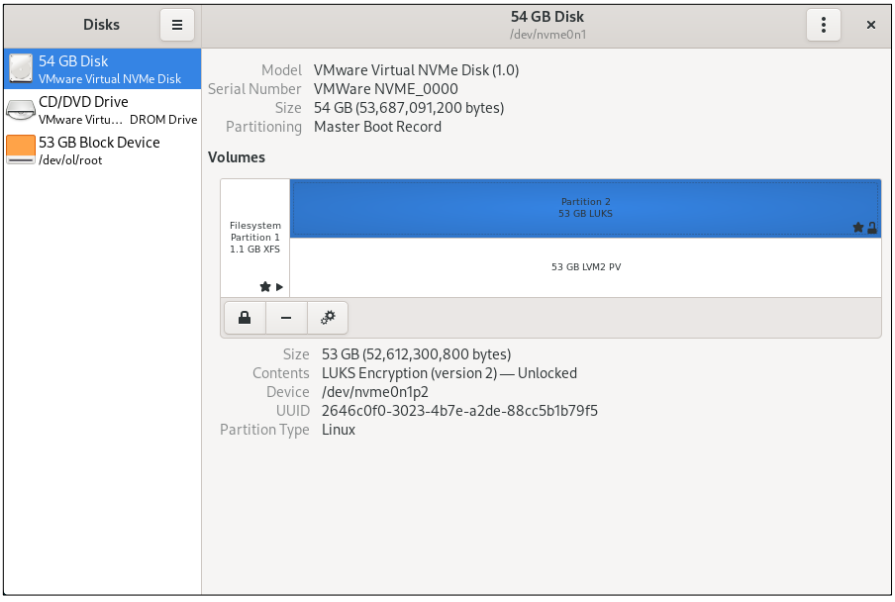

Рисунок 2.3 – Вивід інформації про зашифрований системний розділ /dev/nvme0n1p2

На початковому етапі завантаження операційної системи буде виведено екран запиту паролю для розблокування системного диску та продовження завантаження.

2.2.2 Алгоритми шифрування в LUKS

LUKS використовує кілька криптографічних алгоритмів для забезпечення безпеки шифрування дисків [13]. Основними алгоритмами, що застосовуються в LUKS, є: AES, Serpent та Twofish.

AES є найбільш популярним алгоритмом симетричного блочного шифрування в LUKS. Він використовує різні розміри ключів (128, 192, або 256 біт) і розмір блоку 128 біт. Кількість раундів залежить від довжини ключа: для AES з ключем 128 біт використовується 10 раундів, для ключа 192 біти - 12 раундів, і для ключа 256 біт - 14 раундів. Кожен раунд AES складається з чотирьох основних операцій: SubBytes, ShiftRows, MixColumns і AddRoundKey [14].

Serpent – це ще один алгоритм симетричного блочного шифрування, який використовує розмір блоку 128 біт. Алгоритм Serpent підтримує використання ключів різної довжини - 128, 192 та 256 біт. Він використовує архітектуру мережі заміни-перестановки (SPN) і включає серію з 32 раундів.

Twofish подібний до AES і Serpent. Twofish працює з блоками даних розміром 128 біт та підтримує ключі довжиною від 128 до 256 біт. Алгоритм використовує 16 раундів.

XTS є найпоширенішим режимом роботи для блочного шифрування в LUKS. Він забезпечує високий рівень безпеки та зручність у використанні для шифрування дисків. Цей режим використовує два ключі, зазвичай 256-бітні, та tweak що підвищує стійкість до атак.

Режим CBC менш поширений, але іноді використовується. Він менш захищений від певних видів атак порівняно з XTS, тому використовується рідше.

LUKS використовує майстер-ключ для шифрування даних на диску, а також кілька слотів для зберігання ключів (ключів LUKS), які захищають майстер-ключ. Кожен слот може бути захищений окремим паролем або іншим методом автентифікації. Функція PBKDF2 використовується для обробки паролів користувачів і генерування ключів на основі паролів. Вона застосовує багаторазове хешування пароля разом з сіллю, щоб ускладнити атаки на основі перебору паролів. Argon2i та argon2id може бути використані також як алгоритми PBKDF для отримання ключів з паролів.

Ініціалізація контейнера LUKS відбувається на диску або розділі з вибраним алгоритмом шифрування і режимом роботи. Далі відбувається створення та збереження майстер-ключа, захищеного паролями через слоти ключів LUKS. Дані, що записуються на диск, шифруються майстер-ключем, який захищений алгоритмом шифрування і режимом роботи.

Для доступу до зашифрованих даних користувач вводить пароль, який розшифровує відповідний ключ LUKS, що дозволяє отримати майстер-ключ для розшифрування даних.

Таким чином, LUKS забезпечує потужне та гнучке рішення для шифрування дисків у Linux, використовуючи надійні криптографічні алгоритми та методи управління ключами.

2.3 Шифрування каталогів та файлів

2.3.1 Використання EncFS

EncFS – це метод шифрування, який дозволяє шифрувати окремі файли або каталоги у файловій системі. EncFS працює у просторі користувача і не вимагає спеціальних прав доступу до ядра системи [15]. Нижче наведено кроки для встановлення та налаштування EncFS на Oracle Linux.

Команда `dnf install epel-release` використовується для встановлення репозиторію EPEL, який містить додаткові пакунки для дистрибутиву Oracle

Linux. Встановлення EPEL дозволяє отримати доступ до широкого спектра програмного забезпечення, яке зазвичай не включено в основні репозиторії.

Після встановлення репозиторію EPEL є можливість встановлювати додаткове програмне забезпечення з цього репозиторію. Команда `dnf install encfs` використовується для встановлення пакета fuse-encfs в систему.

На рисунку 2.4 показана детальна інформація про пакет fuse-encfs.

```
[root@oracle9u4 ~]# dnf info fuse-encfs
Last metadata expiration check: 1:56:51 ago on Sat 18 May 2024 06:45:02 PM EEST.
Installed Packages
Name         : fuse-encfs
Version      : 1.9.5
Release      : 12.el9
Architecture : x86_64
Size         : 1.5 M
Source       : fuse-encfs-1.9.5-12.el9.src.rpm
Repository   : @System
From repo    : ol9_developer_EPEL
Summary      : Encrypted pass-thru filesystem in userspace
URL          : http://www.arg0.net/encfs
License      : GPLv3+
Description  : EncFS implements an encrypted filesystem in userspace using FUSE.
             :  FUSE provides a Linux kernel module which allows virtual
             : filesystems to be written in userspace.  EncFS encrypts all data
             : and filenames in the filesystem and passes access through to the
             : underlying filesystem.  Similar to CFS except that it does not
             : use NFS.
```

Рисунок 2.4 – Вивід інформації про пакет fuse-encfs

Пакет fuse-encfs надає можливість створювати зашифровані файлові системи в просторі користувача (userspace) за допомогою технології FUSE [16]. Це означає, що шифрування та розшифрування файлів виконується безпосередньо у користувацькому середовищі, не вимагаючи привілеїв адміністратора.

FUSE - це програмна система, яка дозволяє користувачам створювати власні файлові системи без необхідності редагувати код ядра операційної системи. FUSE надає інтерфейс для створення користувацьких файлових систем, використовуючи модуль ядра, який передає файлові операції з ядра в простір користувача. FUSE дозволяє створювати файлові системи в просторі користувача, що забезпечує більшу безпеку і стабільність, оскільки помилки в

коді файлової системи не призведуть до краху всієї системи. FUSE складається з трьох основних компонентів: модуля ядра, який взаємодіє з файловою системою, бібліотеки користувача, яка обробляє файлові операції та утиліти монтування fusermount, яка використовується для монтування та розмонтування файлових систем, створених за допомогою FUSE. Модуль ядра забезпечує стандартний інтерфейс для файлових систем, що дозволяє обробляти запити файлової системи (створення файлів, читання, запис тощо) і передавати їх у користувацький простір. Бібліотека FUSE, що працює в користувацькому просторі, обробляє ці запити, викликаючи відповідні функції у файловій системі, написаній користувачем.

EncFS шифрує не тільки вміст файлів, але й їхні імена. Це забезпечує додатковий рівень захисту, приховуючи не лише дані, але й структуру каталогів. Пакет fuse-encfs працює, створюючи два каталоги: один для зашифрованих даних і один для розшифрованих даних. Коли користувач працює з розшифрованим каталогом, EncFS автоматично шифрує файли перед збереженням їх у зашифрованому каталозі.

2.3.2 Алгоритми шифрування в EncFS

У EncFS використовуються три різних алгоритми шифрування для захисту даних: AES, Blowfish та Camellia. Кожен з цих алгоритмів має свої особливості та параметри, які впливають на безпеку та ефективність шифрування [14].

Blowfish - це блоковий шифр, який використовується в EncFS. При використання в EncFS його розмір блоку становить 64 біта, а ключі можуть мати довжину від 128 до 256 біт. Кількість раундів впливає на безпеку шифру. Зазвичай, більше раундів означає більшу стійкість до криптоаналізу, але також може вплинути на продуктивність алгоритму. У випадку Blowfish, 16 раундів вважається досить безпечною кількістю.

Camellia - це ще один блоковий шифр, який використовується в EncFS. Він має розмір блоку 128 біт і підтримує ключі довжиною від 128 до 256 біт. Camellia був розроблений японськими криптографами. У шифрі Camellia

використовується 18 раундів шифрування з ключем 128 біт і 24 раунди з ключами 192 або 256 біт..

Усі алгоритми при використання в EncFS підтримують розміри блоку від 64 до 4096 байт.

Загальною метою використання цих алгоритмів є забезпечення високого рівня безпеки та захисту даних у EncFS. Вони дозволяють створювати зашифровані файли та каталоги з даними, які можна безпечно зберігати та передавати, не ризикуючи їхньою конфіденційністю.

2.4 Висновки до розділу

У другому розділі було проведено огляд операційної системи Oracle Linux. Описано методи шифрування збережених даних в операційній системі. Показано можливості LUKS, які дозволяють шифрувати весь диск або окремі розділи. Описано принцип взаємодії LUKS з dm-crypt.

Описано алгоритми шифрування LUKS такі, як AES, Serpent, Twofish в режимах роботи XTS та CBC. Описано функції PBKDF2, argon2i та argon2id, які використовується для обробки паролів користувачів і генерування ключів на основі паролів.

Встановлено пакет fuse-encfs, який надає можливість створювати зашифровані файлові системи в просторі користувача за допомогою технології FUSE. Показано можливості EncFS, які дозволяють шифрувати окремі файли або каталоги у файловій системі. Описано алгоритми шифрування EncFS такі, як AES, Blowfish та Camellia та їх параметри при застосуванні в EncFS.

РОЗДІЛ 3 ТЕСТУВАННЯ МЕТОДІВ ШИФРУВАННЯ В ОПЕРАЦІЙНІЙ СИСТЕМІ ORACLE LINUX

3.1 Процедура шифрування даних за допомогою LUKS

Шифрування даних за допомогою LUKS починається з перевірки встановлення необхідних інструментів, таких як `cryptsetup` [17]. Після цього вибирається диск або розділ, який потрібно зашифрувати. Далі ініціалізується LUKS на вибраному диску або розділі за допомогою команди `cryptsetup luksFormat /dev/<disk>`. Під час процесу буде запропоновано створити пароль для шифрованого тому. Після ініціалізації є можливість відкрити зашифрований том командою `cryptsetup luksOpen /dev/<disk> my_encrypted_volume`, де `my_encrypted_volume` - це ім'я, яке присвоюється відкритому тому.

Наступним кроком є створення файлової системи на відкритому зашифрованому томі, наприклад за допомогою команди `mkfs.ext4 /dev/mapper/my_encrypted_volume`. Після створення файлової системи є можливість змонтувати зашифрований том до певної точки монтування, використовуючи команду `mount /dev/mapper/my_encrypted_volume /mnt/my_mount_point`, де `/mnt/my_mount_point` - це директорія монтування. Тепер є можливість використовувати зашифрований том як звичайний диск: зберігати дані, читати і записувати файли.

Після завершення роботу з зашифрованим томом потрібно відмонтувати його за допомогою команди `umount /mnt/my_mount_point` та закрити зашифрований том командою `cryptsetup luksClose my_encrypted_volume`. Це забезпечує, що дані залишаться зашифрованими і недоступними для неавторизованих користувачів. При наступному доступі до зашифрованого тому буде потрібно повторно відкрити його, ввести пароль і змонтувати його до потрібної точки монтування.

3.1.1 Приклад шифрування USB накопичувача

Для шифрування USB накопичувача за допомогою LUKS потрібно визначити його ідентифікатор за допомогою команди `fdisk -l`. (див. рисунок 3.1)

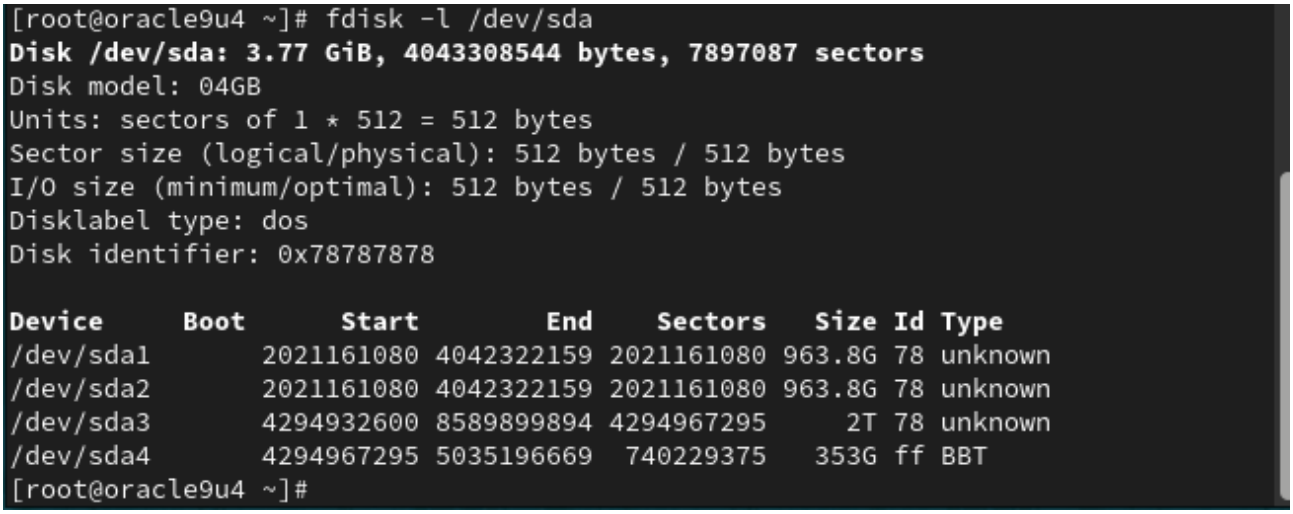

Рисунок 3.1 – Вивід інформації про ідентифікатор USB накопичувача

Команда `fdisk -l` використовується для відображення інформації про всі диски та розділи в системі. Вона надає детальний список підключених пристроїв з інформацією про їх розмір, тип файлової системи та інші атрибути.

Утиліта `cryptsetup` використовується для керування зашифрованими томами. Вона дозволяє створювати, відкривати, закривати, монтувати та розмонтувати зашифровані томи, а також змінювати їх параметри.

Основіні параметри команди `cryptsetup`:

- `luksFormat` - цей параметр використовується для створення нового шифрованого тому у форматі LUKS;

- `luksOpen` - цей параметр відкриває шифрований том, розблоковуючи його для використання;

- `luksClose` - цей параметр закриває відкритий шифрований том;

- `luksDump` - використовується для виведення інформації про LUKS-заголовок, що міститься на пристрої;

- `status` - параметр відображає поточний статус шифрованого тому;

- `benchmark` - параметр використовується для вимірювання швидкості різних методів шифрування;

- `--cipher` - використовується для вказівки алгоритму шифрування, який буде використаний при створенні шифрованого тому з використанням LUKS;

- `--key-size` - використовується для вказівки розміру ключа шифрування при створенні шифрованого тому за допомогою LUKS;

- `--pbkdf` - використовується для встановлення алгоритму функції похідного ключа від паролю, яка застосовується при створенні шифрованого тому з використанням LUKS;

- `--hash` - використовується для вказівки хеш-алгоритму, який буде застосований при створенні похідного ключа від паролю.

На рисунку 3.2 показано команду ініціалізації LUKS на USB накопичувачі.

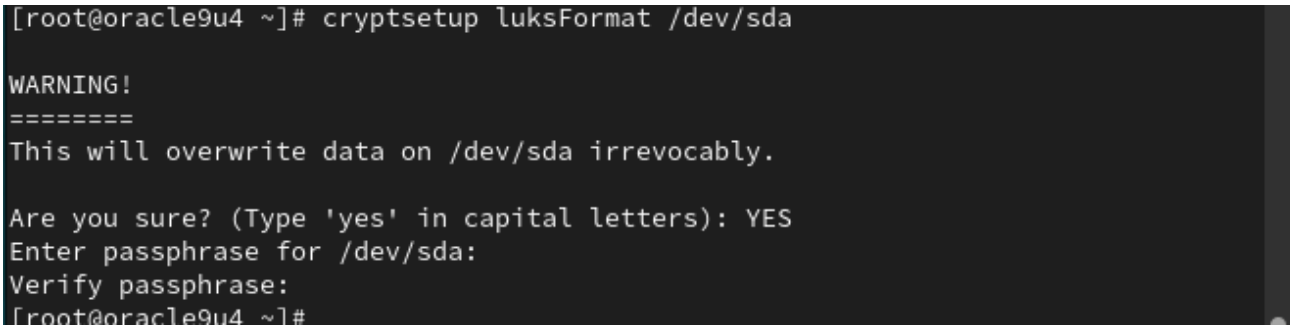

Рисунок 3.2 – Команда ініціалізації LUKS на USB накопичувачі

Користувач повинен ввести парольну фразу для шифрування. Ця парольна фраза буде використовуватися для розблокування зашифрованого диска.

На рисунку 3.3 показано команду для відкриття зашифрованого тому.

Рисунок 3.3 – Команда відкриття зашифрованого тому

Команда створює пристрій у `/dev/mapper` під назвою `encrypted_usb`.

На рисунку 3.4 показано процес форматування розділу з файловою системою ext4 використовуючи утиліту `mkfs.ext4`.

```
[root@oracle9u4 ~]# mkfs.ext4 /dev/mapper/encrypted_usb
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 983039 4k blocks and 245760 inodes
Filesystem UUID: 0d1249eb-e99e-4e02-b5c6-827f46fd3b21
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks):
done
Writing superblocks and filesystem accounting information: done

[root@oracle9u4 ~]#
```

Рисунок 3.4 – Процес форматування зашифрованого розділу

На рисунку 3.5 показано вивід командної оболонки Linux, де виконано команду для підключення шифрованого розділу та перегляду використання дискового простору.

```
[root@oracle9u4 ~]# mount /dev/mapper/encrypted_usb /mnt/usb
[root@oracle9u4 ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  4.0M     0  4.0M   0% /dev
tmpfs                     4.6G     0  4.6G   0% /dev/shm
tmpfs                     1.9G  9.4M  1.9G   1% /run
/dev/mapper/ol-root        49G  6.2G   43G  13% /
/dev/nvme0n1p1            960M  372M  589M  39% /boot
tmpfs                     934M  104K  934M   1% /run/user/1000
/dev/mapper/encrypted_usb  3.7G   24K  3.5G   1% /mnt/usb
[root@oracle9u4 ~]#
```

Рисунок 3.5 – Процес монтування зашифрованого розділу

Команда `mount` використовується для монтування зашифрованого розділу `/dev/mapper/encrypted_usb` до точки монтування `/mnt/usb`. Зашифрований розділ тепер доступний для використання в системі за шляхом `/mnt/usb`.

На рисунку 3.6 показано вивід команди для перевірки інформації про блоковий пристрій /dev/sda.

```
[root@oracle9u4 ~]# blkid /dev/sda
/dev/sda: UUID="295cfcb3-3e7c-495b-988d-6f67cb981fd8" TYPE="crypto_LUKS"
[root@oracle9u4 ~]#
```

Рисунок 3.6 – Вивід інформації про блоковий /dev/sda

Унікальний ідентифікатор для даного розділу UUID використовується системою для однозначного визначення розділів незалежно від їх монтування та фізичного розташування. Тип crypto_LUKS вказує, що розділ зашифровано за допомогою LUKS.

На рисунку 3.7 показано вивід команди для перегляду деталей шифрування та налаштування LUKS.

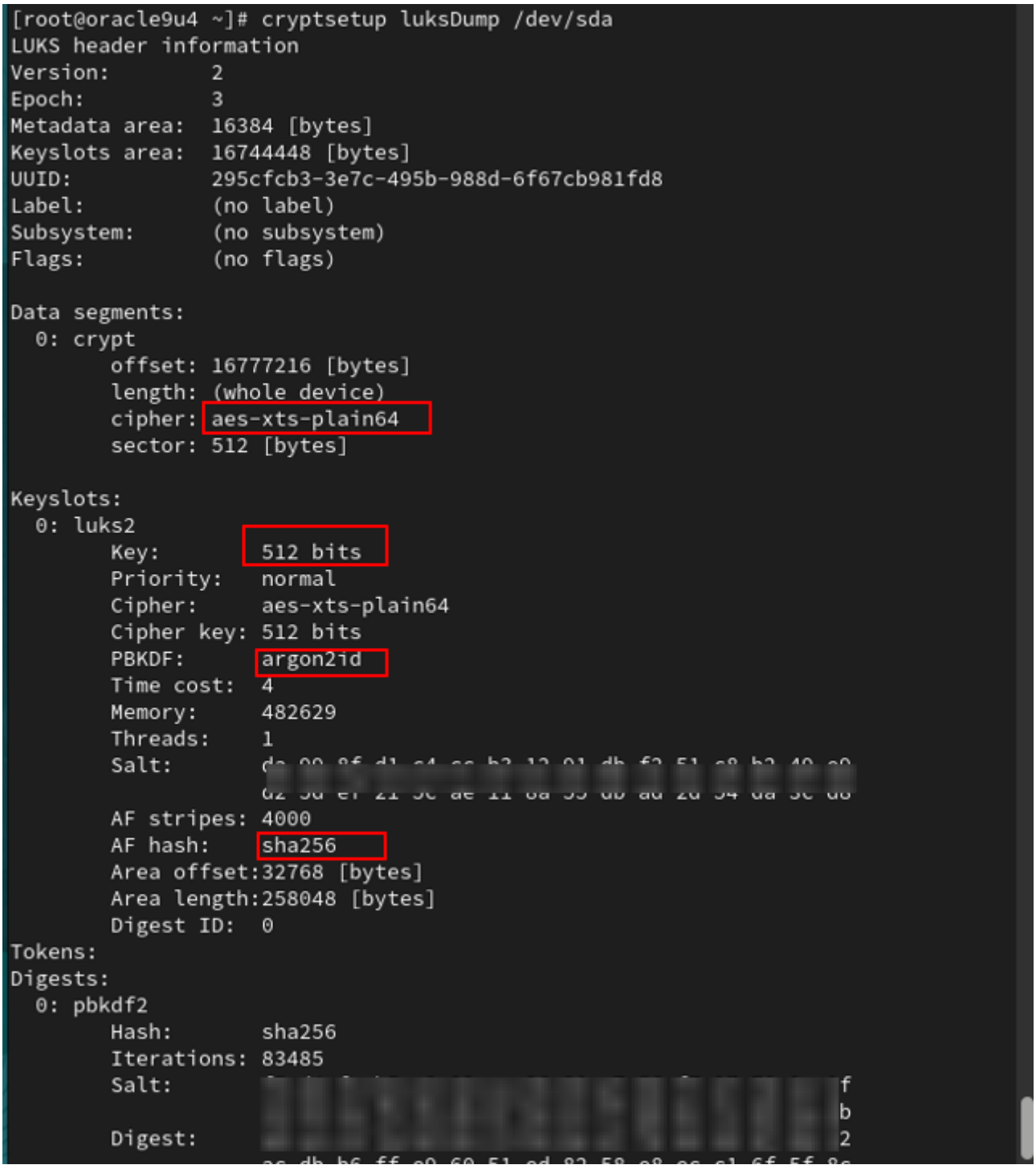

Рисунок 3.7 – Вивід команди перегляду деталей шифрування /dev/sda

В виводі команди можна побачити деталі шифрування та налаштування LUKS. Використовується алгоритм шифрування aes-xts-plain64 і розмір сектора 512 байтів. Слот ключів 0 належить до типу luks2, має розмір ключа 512 біт, нормальний пріоритет, використовує алгоритм шифрування aes-xts-plain64, має розмір ключа шифрування 512 біт, використовує алгоритм похідного ключа від паролю argon2id, сіль для PBKDF в шістнадцятковому форматі, хеш-алгоритм для AF - sha256, ідентифікатор дайджесту 0. Дайджест 0 використовує хеш-алгоритм sha256, має 83485 ітерацій, сіль для хешу у шістнадцятковому форматі, сам хеш-дайджест у шістнадцятковому форматі. Алгоритм шифрування aes-xts-plain64 забезпечує високу стійкість до атак. Алгоритм похідного ключа від паролю argon2id є сучасним та надійним, стійким до атак перебором. Розмір ключа у 512 біт вказує на високий рівень захисту. Сіль використовується для захисту від атак на основі заготовлених таблиць хешів. Велика кількість ітерацій забезпечує додатковий рівень безпеки. Ці параметри разом забезпечують високий рівень захисту для даних, використовуючи сучасні алгоритми шифрування і функції похідного ключа від паролю.

3.1.2 Зміна параметрів шифрування

Для зміни параметрів шифрування, таких як алгоритм шифрування, функція похідного ключа або хеш-алгоритм, потрібно створити новий LUKS-том з новими параметрами і перемістити дані на нього.

Проведемо шифрування USB накопичувачі з явно встановленими параметрами. Спочатку потрібно відмонтувати та закрити зашифрований том (див.рисунок 3.8).

```
[root@oracle9u4 ~]# umount /mnt/usb
[root@oracle9u4 ~]# cryptsetup luksClose encrypted_usb
```

Рисунок 3.8 – Відмонтування та закриття зашифрованого тому

На рисунку 3.9 показана команда ініціалізації LUKS на USB накопичувачі з явно встановленими параметрами шифрування.

```
[root@oracle9u4 ~]# cryptsetup luksFormat --pbkdf pbkdf2  --hash ripemd160 --key
-size 256 --cipher serpent-xts-plain64 /dev/sda --debug
```

Рисунок 3.9 – Ініціалізація LUKS з явно встановленими параметрами шифрування

Команда створює новий LUKS-том на пристрої /dev/sda з використанням функції похідного ключа від паролю PBKDF2, хеш-алгоритму RIPEMD-160, розміру ключа 256 біт, алгоритму шифрування Serpent у режимі XTS. Процес виконання команди наведено в додатку А.

На рисунку 3.10 показано вивід команди для перегляду деталей шифрування та налаштування LUKS при встановлення параметрів шифрування явно.

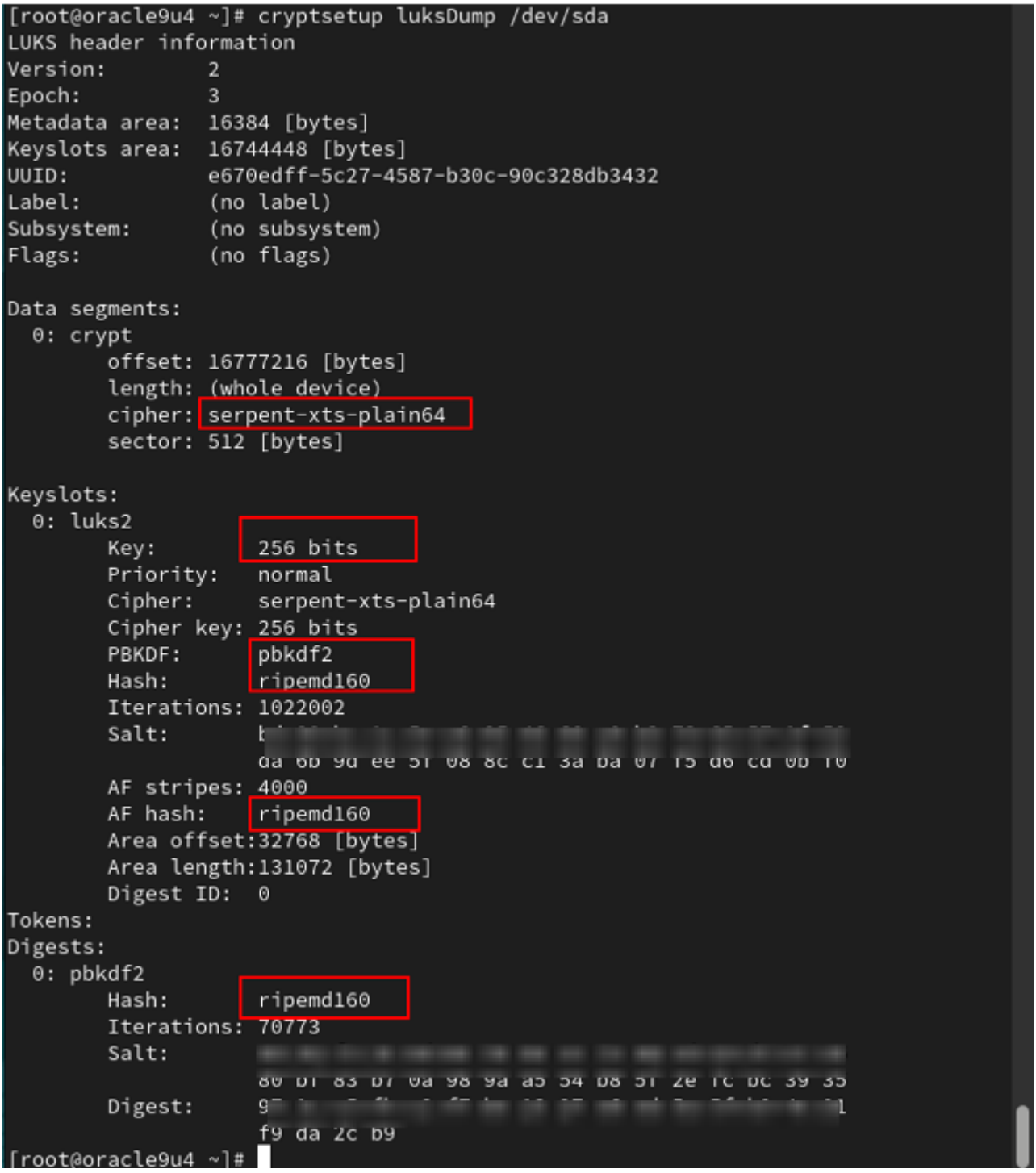

Рисунок 3.10 – Вивід команди перегляду деталей шифрування /dev/sda

Вивід показує, що використовується алгоритм шифрування serpent-xts-plain64 та розмір сектора 512 байт. Параметр plain64 визначає схему генерації вектора ініціалізації IV. Він вказує, що для кожного блоку даних IV генерується як 64-бітне значення, засноване на позиції блоку. Цей підхід забезпечує унікальний IV для кожного блоку, зменшуючи ризик повторення IV, що може призвести до криптографічної слабкості.

Слот ключів 0 включає тип слота luks2, розмір ключа 256 біт, алгоритм шифрування serpent-xts-plain64, розмір ключа шифрування 256 біт, функцію похідного ключа pbkdf2, хеш-алгоритм ripemd160, кількість ітерацій для PBKDF2 -1022002, сіль у шістнадцятковому форматі. Дайджест 0 включає хеш-алгоритм ripemd160, кількість ітерацій - 70773, сіль у шістнадцятковому форматі та значення хешу у шістнадцятковому форматі.

3.2 Процедура шифрування даних за допомогою EncFS

Шифрування EncFS забезпечується на рівні окремих файлів, що дозволяє гнучко управляти доступом до зашифрованих файлів і здійснювати синхронізацію через мережу без потреби в шифруванні цілого диска [16].

Спочатку створюється пара каталогів. Один для зберігання зашифрованих даних і інший, який є точкою монтування, де файли з'являються у незашифрованому вигляді після введення пароля. При доступі до файлів через точку монтування EncFS динамічно шифрує або розшифровує дані на льоту.

Процес шифрування з EncFS включає вибір пароля, який використовується для генерації ключа шифрування. Цей ключ застосовується для шифрування вмісту файлів перед їхнім зберіганням у зашифрованому каталозі. Ключі та конфігураційні параметри зберігаються у зашифрованому вигляді у каталозі, тому доступ до файлів можливий лише після автентифікації користувача.

Для використання EncFS достатньо просто скопіювати, змінити або видалити файли у точці монтування, як це робиться зі звичайними файлами. Це забезпечує високий рівень зручності, тому що не потрібно вручну шифрувати чи розшифровувати кожен файл. EncFS ідеально підходить для захисту конфіденційних даних на портативних пристроях або в хмарних сховищах, оскільки він забезпечує безпеку даних без необхідності шифрування всього диска і дозволяє легко синхронізувати зашифровані файли між пристроями.

3.2.1 Приклад шифрування каталогу

Створимо на USB накопичувачі файлову систему FAT. Ця файлова система, сумісна з Windows, та зазвичай використовується для зовнішніх носіїв або розділів, які потрібно читати в різних операційних системах.

На рисунку 3.11 показано команди, які використовується для форматування диска /dev/sda у файлову систему FAT.

```
[root@oracle9u4 ~]# mkfs.vfat /dev/sda
mkfs.fat 4.2 (2021-01-31)
[root@oracle9u4 ~]# fdisk -l /dev/sda
Disk /dev/sda: 3.77 GiB, 4043308544 bytes, 7897087 sectors
Disk model: 04GB
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

Рисунок 3.11 – Створення файлової системи FAT на USB накопичувачі

Щоб створити зашифровану файлову систему на USB накопичувачі, потрібно визначити два каталоги: один для зберігання зашифрованих файлів `/mnt/usb/encrypted`, інший для монтування і доступу до розшифрованих файлів `/mnt/usb/decrypted` (див.рисунок 3.12).

```
[root@oracle9u4 ~]# mkdir /mnt/usb/encrypted
[root@oracle9u4 ~]# mkdir /mnt/usb/decrypted
[root@oracle9u4 ~]# ls -l /mnt/usb/
total 8
drwxr-xr-x. 2 root root 4096 May 21 18:37 decrypted
drwxr-xr-x. 2 root root 4096 May 21 18:37 encrypted
[root@oracle9u4 ~]#
```

Рисунок 3.12 – Створення тестових каталогів

На рисунку 3.13 показано запуску EncFS для ініціалізації нової зашифрованої файлової системи зі стандартним режимом шифрування.

```
[root@oracle9u4 ~]# encfs --standard /mnt/usb/encrypted /mnt/usb/decrypted
Creating new encrypted volume.
Standard configuration selected.

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: 192 bits
Block Size: 1024 bytes
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.
File holes passed through to ciphertext.

Now you will need to enter a password for your filesystem.
You will need to remember this password, as there is absolutely
no recovery mechanism.  However, the password can be changed
later using encfsctl.

New Encfs Password:
Verify Encfs Password:
[root@oracle9u4 ~]#
```

Рисунок 3.13 – Створення віртуальної зашифрованої файлової системи

Після введення паролю для шифрування, ініціалізації і монтування зашифрованого каталогу `/mnt/usb/encrypted` є можливість працювати з файлами у каталозі `/mnt/usb/decrypted`, як з будь-якою іншою файловою системою (див.рисунок 3.14).

```
[root@oracle9u4 ~]# df -h
Filesystem           Size  Used Avail Use% Mounted on
devtmpfs             4.0M     0  4.0M   0% /dev
tmpfs                4.6G     0  4.6G   0% /dev/shm
tmpfs                1.9G  9.4M  1.9G   1% /run
/dev/mapper/ol-root   49G  6.2G   43G  13% /
/dev/nvme0n1p1       960M  372M  589M  39% /boot
tmpfs                934M  104K  934M   1% /run/user/1000
/dev/sda             3.8G   16K  3.8G   1% /mnt/usb
encfs                3.8G   16K  3.8G   1% /mnt/usb/decrypted
[root@oracle9u4 ~]#
```

Рисунок 3.14 – Змонтована шифрована файлова система

Файли, які копіюються або створюються в каталозі `/mnt/usb/decrypted`, автоматично шифруються і зберігаються в каталозі `/mnt/usb/encrypted`. Створимо в каталозі `/mnt/usb/decrypted` текстовий файл `testencfs.txt` з довільним вмістом (див.рисунок 3.15).

```
[root@oracle9u4 ~]# cd /mnt/usb/decrypted/
[root@oracle9u4 decrypted]# touch ./testencfs.txt
[root@oracle9u4 decrypted]# echo "Test EncFS 21.05.2024 TNTU" > ./testencfs.txt
[root@oracle9u4 decrypted]#
```

Рисунок 3.15 – Створення текстового файлу `testencfs.txt`

В каталозі `/mnt/usb/encrypted` було збережено зашифрований файл `testencfs.txt` (див.рисунок 3.16).

```
[root@oracle9u4 encrypted]# ls -la
total 16
drwxr-xr-x. 2 root root 4096 May 21 18:59 .
drwxr-xr-x. 4 root root 4096 Jan  1  1970 ..
-rwxr-xr-x. 1 root root   35 May 21 18:59 ,9mrOiwTIiQvKs9TwxgZT5t9
-rwxr-xr-x. 1 root root 1297 May 21 18:45 .encfs6.xml
[root@oracle9u4 encrypted]#
```

Рисунок 3.16 – Зашифрований текстовий файл `testencfs.txt`

Файл `.encfs6.xml` є конфігураційним файлом EncFS, який зберігається у зашифрованому каталозі. Цей файл містить усі налаштування і параметри шифрування, які були задані під час створення зашифрованої файлової системи. Оскільки цей файл містить важливі криптографічні метадані, його втрата означатиме втрату доступу до зашифрованих даних.

На рисунку 3.17 показано вміст файлу `.encfs6.xml`.

Рисунок 3.17 – Вміст файлу `.encfs6.xml`

Файл використовує XML-версію 1.0 з кодуванням UTF-8. Конфігурація зберігається у тегу <cfg>. В межах цього тегу зберігаються всі налаштування шифрування. Версія конфігурації 20100713 та версія EncFS 1.9.5 зберігаються у тегах <version> і <creator> відповідно. Алгоритм шифрування визначений як ssl/aes з версією 3.0, що вказує на використання алгоритму AES через OpenSSL.

Алгоритм іменування файлів визначений як nameio/block з версією 4.0. Це вказує на використання блочного алгоритму для шифрування імен файлів. Розмір ключа для шифрування становить 192 біти, а розмір блоку – 1024 байти. Параметр plainData встановлений на 0, що вказує на те, що дані не зберігаються

у відкритому вигляді. Параметр uniqueIV встановлений на 1, що означає використання унікального вектора ініціалізації (IV) для кожного файлу. Параметр chainedNameIV також встановлений на 1, що вказує на використання пов'язаних IV для імен файлів.

Довжина солі становить 20 байтів. Сіль зберігається у тегу <saltData> як закодований у base64 рядок. Кількість ітерацій для функції похідного ключа (KDF) становить 457262, а бажана тривалість KDF становить 500 мілісекунд.

Ці параметри разом визначають, як EncFS шифрує і дешифрує файли у цьому зашифрованому каталозі.

3.2.2 Зміна параметрів шифрування

Зміна параметрів шифрування EncFS включає створення нового зашифрованого каталогу з бажаними налаштуваннями та перенесення даних зі старого каталогу.

Щоб створити зашифровану файлову систему на USB накопичувачі, потрібно знову визначити два каталоги: один для зберігання зашифрованих файлів `/mnt/usb/encrypted2`, інший для монтування і доступу до розшифрованих файлів `/mnt/usb/decrypted2` та запустити EncFS для ініціалізації нової зашифрованої файлової системи зі зміненими параметрами шифрування. (див.рисунок 3.18).

```
[root@oracle9u4 encrypted]# mkdir /mnt/usb/encrypted2 && mkdir /mnt/usb/decrypte
d2
[root@oracle9u4 encrypted]# encfs  /mnt/usb/encrypted2 /mnt/usb/decrypted2
```

Рисунок 3.18 – Створення тестових каталогів для нової зашифрованої файлової системи

Процес створення нової зашифрованої файлової системи зі зміненими параметрами шифрування наведено в додатку Б.

На рисунку 3.19 показано вивід команди `encfsctl info`, яка надає детальну інформацію про конфігурацію зашифрованої файлової системи EncFS, зокрема

алгоритм шифрування, розмір ключа, параметри PBKDF2, розмір блоку та інші важливі налаштування.

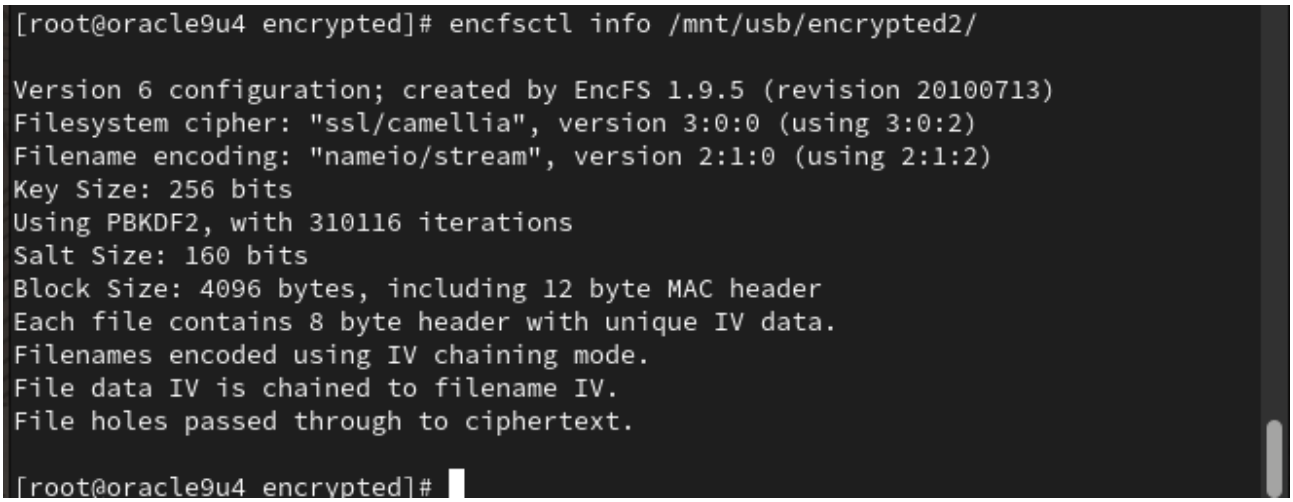

Рисунок 3.19 – Вивід команди `encfsctl info`

Алгоритм шифрування файлової системи визначений як ssl/camellia. Для шифрування імен файлів використовується алгоритм nameio/stream. Розмір ключа шифрування становить 256 біт. Для отримання ключа використовується алгоритм PBKDF2 з 310116 ітераціями. Розмір значення солі становить 160 біт. Розмір блоку даних складає 4096 байт, включаючи 12 байтів MAC заголовка. Кожен файл містить 8 байт заголовка з унікальними даними IV. Імена файлів кодуються з використанням режиму IV chaining. IV даних файлу пов'язаний з IV імені файлу. Пробіли у файлах передаються до шифротексту.

Ця інформація забезпечує розуміння того, як саме EncFS шифрує дані та імена файлів у цій зашифрованій файловій системі.

3.2.3 Сумісність шифрування

EncFS в Linux створює віртуальний зашифрований том, який теоретично може працювати в будь-якій операційній системі, включаючи Windows. Проте EncFS розроблений спеціально для Linux-систем і офіційно не підтримується на Windows.

Це означає, що використання EncFS на Windows можливе, але потребує додаткових зусиль і інструментів для налаштування.

EncFSMP - це програма, яка дозволяє використовувати зашифровані томи EncFS на Windows [18]. Вона забезпечує користувачам можливість монтувати і працювати з зашифрованими даними, створеними в Linux, без необхідності додаткових налаштувань.

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

На рисунках 3.20 - 3.23 показано порядок розшифрування в операційній системі Windows 10 вмісту каталогу, який зашифрований в операційній системі Oracle Linux за допомогою EncFS.

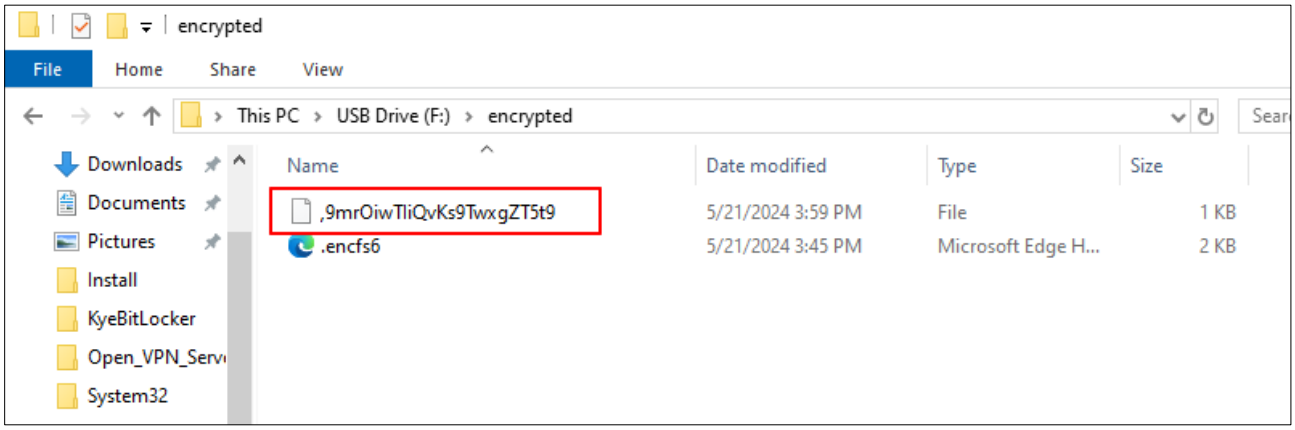

Рисунок 3.20 – Зашифрований файл `testencfs.txt`

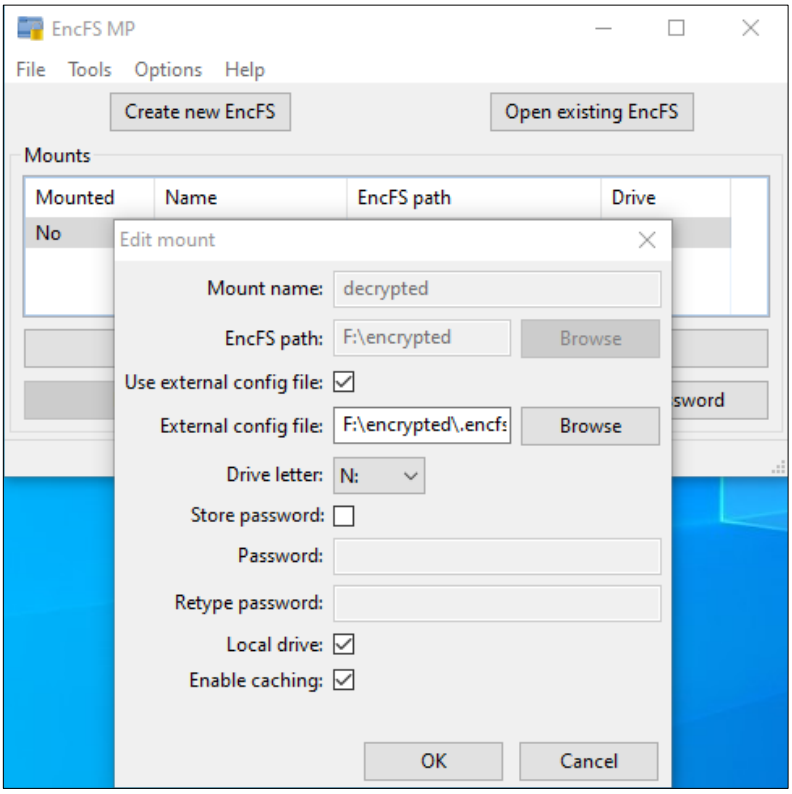

Рисунок 3.21 –Параметри розшифрування вмісту encrypted каталогу

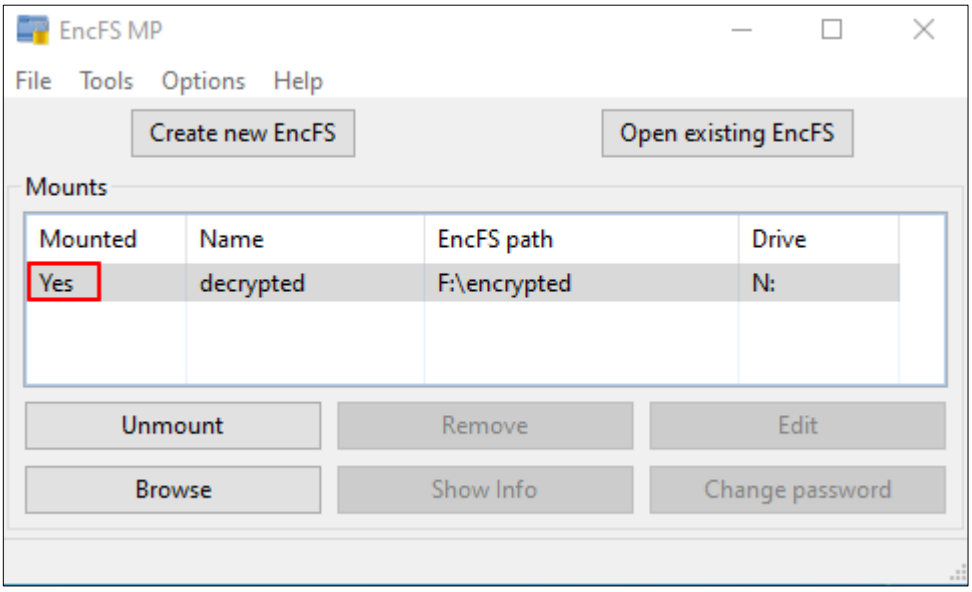

Рисунок 3.22 – Розшифрований та змонтований як диск N вміст encrypted каталогу

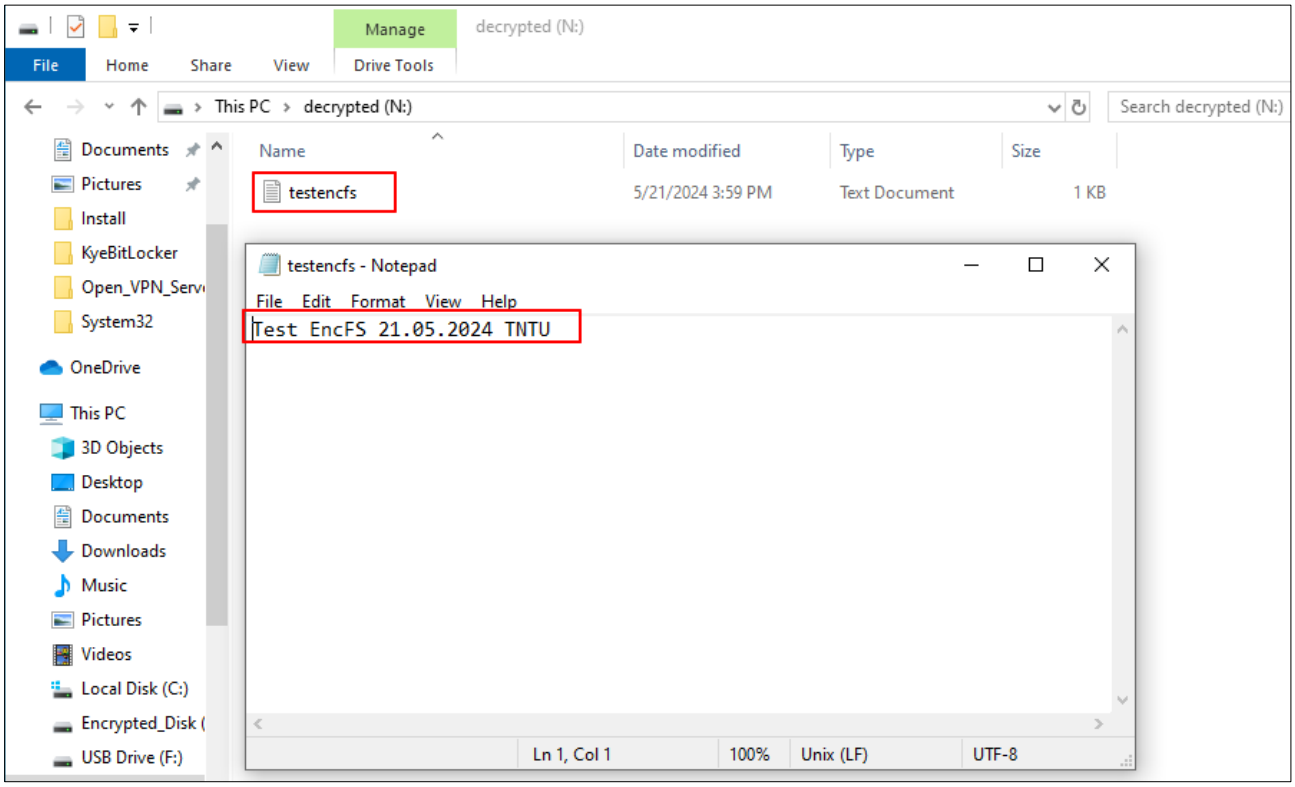

Рисунок 3.23 – Вміст диску N та розшифрований вміст файлу `testencfs.txt`

EncFSMP надає простий графічний інтерфейс, що робить процес роботи з зашифрованими томами більш зручним і доступним для користувачів Windows. Однак, EncFSMP може мати обмежену функціональність у порівнянні з версією EncFS для Linux і не завжди забезпечує той самий рівень стабільності та безпеки.

3.3 Висновки до розділу

В третьому розділі було проведено процедуру шифрування даних за допомогою LUKS на прикладі шифрування USB накопичувача. Показано можливості утиліти cryptsetup та описано параметри шифрування. Докладно описано процес зміни алгоритму шифрування та параметрів шифрування LUKS.

Показано процедуру шифрування даних за допомогою EncFS. Показано можливості утиліти encfs та описано параметри шифрування. Проведено опис процесу зміни алгоритму шифрування та параметрів шифрування EncFS. Показано можливості утиліти encfsctl. Описано можливості програми EncFSMP, яка дозволяє використовувати зашифровані томи EncFS в Windows.

Продемонстровано порядок розшифрування в операційній системі Windows вмісту каталогу, який зашифрований EncFS в операційній системі Linux.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом

У сучасному світі електрична енергія є невід'ємною частиною нашого повсякденного життя. Вона живить наші побутові прилади, освітлює наші будинки та забезпечує роботу промислових комплексів. Однак, не зважаючи на всі переваги, пов'язані з електричною енергією, вона також може стати причиною небезпеки для людей.

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 [19].

Ураження електричним струмом може мати серйозні наслідки для здоров'я і навіть призвести до смерті. У таких випадках надання долікарської допомоги має вирішальне значення і може врятувати життя потерпілого.

Одразу після того, як особа отримала ураження електричним струмом, важливо негайно забезпечити безпеку для особи яка буде надавати долікарську допомогу та навколишніх людей, які будуть відігравати другорядну роль, адже при близькому контакті з прямим джерелом електронебезпеки є висока вірогідність отримання травм. Перш за все, необхідно відключити джерело електричного струму. Найкращим вирішенням проблеми буде вимкнення перемикача або витягнення вилки з розетки. Важливо не торкатися постраждалої особи без наявності ізольованих рукавичок або інших засобів ізоляції, також необхідно розміститись на сухій та ізольованій поверхні. Пересування особи без крайньої на це потреби є суворо заборонене. Наступним кроком є виклик медичної допомоги. Швидка допомога має бути повідомлена про ураження електричним струмом і про наявність потерпілого інформація про стан потерпілого повинна надаватися швидко і чітко. Тимчасова втрата свідомості, опіки або інші видимі пошкодження шкіри можуть свідчити про серйозні

ураження, тому необхідно негайно звернутися за медичною допомогою, це значно збільшить шанси постраждалого на виживання.

Важливим етапом долікарської допомоги є оцінка стану потерпілого. При виявленні відсутності дихання, необхідно розпочати штучне дихання та серцево-легеневу реанімацію (СЛР). Навчитися правильно проводити СЛР – це надзвичайно важлива навичка, яку слід мати кожному, оскільки можливість врятувати життя людини залежить від швидкості та належного надання першої медичної допомоги. Якщо потерпілий дихає, але не має пульсу, слід провести тільки СЛР.

Далі, слід перевірити наявність інших травм або пошкоджень, таких як переломи або опіки. Якщо на тілі є виявлено опіки, необхідно негайно охолодити пошкоджену ділянку. Холод допоможе зменшити біль і запобігти подальшому ушкодженню тканин. Важливо уникати накладання льоду безпосередньо на шкіру, оскільки це може призвести до обмороження.

Ураження електричним струмом може також спричинити серцеві порушення, такі як фібриляція шлуночків. У такому випадку використання автоматизованого зовнішнього дефібрилятора (АЗД) може бути вирішальним. АЗД - це пристрій, який дозволяє провести електричний розряд через грудну клітку, щоб відновити нормальний ритм серця. Цей пристрій широко розповсюджений в публічних місцях, таких як аеропорти, торгові центри та інші громадські приміщення. Знання про розташування найближчого АЗД може врятувати життя у разі ураження електричним струмом.

Важливою частиною долікарської допомоги є попередження додаткових ушкоджень. Необхідно переконатися, що потерпілий знаходиться в безпечному місці, де немає ризику подальших уражень. Також необхідно уникати контакту з вологою поверхнею, оскільки вода може проводити електричний струм. Якщо потерпілий перебуває в небезпеці, слід негайно повідомити про це спеціалістів із рятувальних служб. Однак, найкращим рішенням є попередження уражень електричним струмом. Необхідно дотримуватися всіх правил безпеки, пов'язаних з роботою з електричними пристроями. Категорично заборонено ремонтувати пошкоджену електричну розетку або провід, без наявності

необхідних знань і навичок. Використання ізольованих рукавичок та інструментів, при роботі з електричними проводами є надзвичайно важливим. Установка розеток з диференційними автоматами (РДА), які автоматично відключають електропостачання при виявленні небезпечних ситуацій дозволить запобігти виникненню екстрених ситуацій.

Ураження електричним струмом є серйозною небезпекою для здоров'я і може мати фатальні наслідки. Долікарська допомога, надана на ранніх етапах, може врятувати життя потерпілого. Важливо знати основні принципи надання першої медичної допомоги у разі ураження електричним струмом і дотримуватися всіх правил безпеки під час роботи з електричними приладами [20]. Запобігання ураженням електричним струмом є найкращим підходом, але знання про першу допомогу у разі виникнення небезпеки дозволять зберегти найцінніше – людське життя та здоров’я.

4.2 Вплив шуму на організм людини та розробка заходів щодо його зниженню до допустимих величин для обладнання

Під шумом розуміють набір багаточисельних звуків, які швидко змінюються за частотою, силою і складаються з ряду гармонік [21]. З фізичної точки зору звуки є механічними коливальними рухами частинок пружного середовища в діапазоні частот, що чує людина. Звукові гармоніки розповсюджуються у вигляді хвиль.

Шум є загально-біологічним подразником, діє не тільки на органи слуху, але може викликати порушення роботи серцево-судинної і нервової систем, зумовлювати професійні захворювання [21]. Основними характеристиками звукових коливань є інтенсивність (сила), частота і форма звукової хвилі. Інтенсивність визначається енергією, що переноситься за 1 с звуковою хвилею через поверхню площею 1 м^2, яка перпендикулярна напрямку розповсюдження звукової хвилі. Діапазон тисків, що сприймає вухо людини, дуже широкий (10-12Вт/м2 – поріг больового відчуття, верхня межа) [21].

З розвитком промисловості все більший контингент людей підпадає під вплив вібрацій, які являють собою механічні коливання, що передаються тілу людини. Основні параметри вібрацій – частота та амплітуда коливань, але на відміну від шуму, при якому енергія механічних коливань передається через повітряне середовище, при дії вібрацій вона розповсюджується по тканинах і викликає їх коливання або тіла людини в цілому [21]. Найбільш небезпечна вібрація частотою 16-250 Гц, дія якої призводить до вібраційної хвороби.

Нормування шуму здійснюється згідно з “Санітарними нормами допустимих рівнів шуму на робочих місцях”. В Україні застосовується принцип нормування шуму на основі граничних спектрів (гранично допустимих рівнів звукового тиску) в октавних смугах частот та еквівалентних рівнів звуку. Гранично-допустимі рівні шумів санітарними нормами встановлені для кожного класу [18]:

– для високочастотних шумів (вище 800 Гц) – 75-85 дБ;

– для середньо частотних шумів (300-800 Гц) – 85-90 дБ;

– для низькочастотних шумів (до 300 Гц) – 90-100 дБ.

Шумові явища мають якість кумуляції, накопичуючись в організмі, вони все більше і більше пригнічують нервову систему. Відомо, що після шумової дії інтенсивністю 120 дБ протягом однієї години потрібно 5 годин, щоб гострота слуху повернулась до норми. Стабільні широкосмугові шуми, які перевищують граничний рівень, викликають зниження темпу, ефективності й якості роботи операторів [21].

Ультразвук широко застосовують у технологічних процесах виготовлення радіоелектронної апаратури (промивка деталей, зварювання мініатюрних вузлів тощо) з частотою вище 2220 кГц. При цьому густина енергії ультразвукових коливань у мільйони разів більша густини енергії звуків, які ми чуємо. Тому під його дією відбувається нагрівання тіла, а при дії коливань через рідкі і тверді середовища відбувається розривання і руйнування тканин [21]. Захист від ультразвуку, який діє через повітряне середовище, досягається шляхом звукоізоляції установок (листова сталь, дюралюміній, що обклеєні гумою або руберойдом, гетинакс) або розміщення їх в окремій звукоізолюючій кабіні.

Ультразвукові установки повинні мати блокування, яке відключає генератор ультразвукових коливань в момент відкривання кришок або кожухів [21]. Для запобігання шкідливої дії шуму і вібрації на організм працюючих проводяться технічні, організаційні і медико-профілактичні заходи. Одним з основних технічних заходів є зменшення при експлуатації та на стадії проектування, конструювання обладнання причин шуму і вібрації в самому джерелі утворення. Досягають цього завдяки використанню раціональної конструкції обладнання, заміни ударної дії деталей і машин коливальною, з'єднання елементів гнучкими зв'язками, врівноважування обертових частин механізмів, заміни металевих деталей пластмасовими, забезпечення різних власних частот коливань механізму з частотою збуджуючої сили [21]. Якщо неможливо ізолювати чи знизити шум і вібрацію самого джерела, потрібно:

– ізолювати джерело шуму або вібрації від навколишнього середовища засобами вібро- та звукоізоляції;

– раціонально планувати виробничі приміщення, що мають інтенсивні джерела шуму;

– збільшувати звукопоглинання внутрішніх поверхонь приміщення шляхом звукопоглинальних покриттів.

Якщо не вдається зменшити рівень шуму і вібрації на робочому місці до нормативних значень та необхідно використовувати засоби індивідуального захисту: рукавиці, взуття, навушники, м'які шоломи, які зменшують рівень звукового тиску на 40-50 дБ.

У процесі виробництв, експлуатації і зберігання комп'ютерної і радіоелектронної апаратури можуть виникати механічні і динамічні дії, що характеризуються широким діапазоном частот коливань, а також амплітудою, прискоренням і часом дії [21].

При експлуатації високочастотного обладнання всередині виробничих приміщень зниження напруженості електромагнітного випромінювання досягається такими методами:

– захист часом – обмеження часу перебування людини в електромагнітному полі, що залежить від інтенсивності опромінення або напруженості ЕМП.

– захист відстанню застосовується при неможливості послабити інтенсивність опромінення в заданій зоні іншими методами: збільшують відстань між джерелом випромінювання і обслуговуючим персоналом;

– добре виконане екранування джерела і усунення нещільності у фланцевих з’єднаннях, фідерів, зазорів у обшивці корпусів, нещільних електричних контактів;

– проведення дистанційного контролю й управління роботою передавачів з екранованого приміщення;

– засобами індивідуального захисту.

В залежності від типу джерела випромінювання, його потужності, характеру технологічного процесу може застосовуватись один з вказаних методів або будь-яка їх комбінація.

ВИСНОВКИ

У процесі написання кваліфікаційної роботи було проаналізовано та продемонстровано практичну реалізацію методів шифрування збережених даних в операційній системі Oracle Linux.

У першому розділі проведено огляд криптографічних функцій та методів. Детально розглянуто симетричне шифрування, його переваги та недоліки. Описано режими роботи блокового шифрування: ECB, CBC, CFB, OFB, CTR, GCM та XTS. Проведено порівняння блокового та потокового симетричного шифрування. Також детально розглянуто асиметричне шифрування, включаючи принципи використання відкритого та приватного ключів в асиметричному шифруванні та цифрових підписах. Розглянуто асиметричні шифри RSA та ECC. Крім того, описано принцип роботи та можливості застосування хеш-функцій і генератора псевдовипадкових чисел.

У другому розділі проведено огляд операційної системи Oracle Linux. Описано методи шифрування збережених даних в цій операційній системі. Розглянуто можливості LUKS, що дозволяють шифрувати весь диск або окремі розділи. Детально описано принцип взаємодії LUKS з dm-crypt. Описано алгоритми шифрування, які використовує LUKS, зокрема AES, Serpent і Twofish, у режимах роботи XTS та CBC. Також описано функції PBKDF2, argon2i та argon2id, що використовуються для обробки паролів користувачів та генерації ключів на основі паролів. Встановлено пакет fuse-encfs, який надає можливість створювати зашифровані файлові системи в просторі користувача за допомогою технології FUSE. Показано можливості EncFS, що дозволяють шифрувати окремі файли або каталоги у файловій системі. Описано алгоритми шифрування EncFS, зокрема AES, Blowfish та Camellia, а також їх параметри при застосуванні в EncFS.

У третьому розділі досліджено процес шифрування даних з використанням LUKS на прикладі шифрування USB-накопичувача. Продемонстровано можливості утиліти cryptsetup та розглянуто параметри шифрування. Окремо розглянуто процедуру зміни алгоритму шифрування та параметрів шифрування

в LUKS. Проведено шифрування даних за допомогою EncFS. Розглянуто можливості утиліти encfs та надано опис параметрів шифрування. Показано процес зміни алгоритму шифрування та параметрів у EncFS. Детально розглянуто можливості утиліти encfsctl. Описано можливості програми EncFSMP, яка забезпечує підтримку зашифрованих томів EncFS в операційній системі Windows. Продемонстровано процес розшифрування в операційній системі Windows вмісту каталогу, зашифрованого EncFS в операційній системі Linux.

Дослідження та практична реалізація засвідчили ефективність і надійність методів шифрування даних у стані спокою в операційній системі Oracle Linux.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Cryptography URL: https://www.baeldung.com/cs/introduction-to-cryptography (дата звернення: 23.05.2024).
2. Тимощук, В., Долінський, А., & Тимощук, Д. (2024). ЗАСТОСУВАННЯ ГІПЕРВІЗОРІВ ПЕРШОГО ТИПУ ДЛЯ СТВОРЕННЯ ЗАХИЩЕНОЇ ІТ-ІНФРАСТРУКТУРИ. Матеріали конференцій МЦНД, (24.05. 2024; Запоріжжя, Україна), 145-146. https://doi.org/10.62731/mcnd-24.05.2024.001
3. Symmetric Cryptography vs Asymmetric Cryptography. URL: https://www.baeldung.com/cs/symmetric-vs-asymmetric-cryptography (дата звернення: 23.05.2024).
4. Karnaukhov, A., Tymoshchuk, V., Orlovska, A., & Tymoshchuk, D. (2024). USE OF AUTHENTICATED AES-GCM ENCRYPTION IN VPN. Матеріали конференцій МЦНД, (14.06. 2024; Суми Україна), 191-193.
5. Differences Between Stream Cipher and Block Cipher URL: https://www.baeldung.com/cs/stream-cipher-vs-block-cipher (дата звернення: 23.05.2024).
6. Тимощук, В., & Стебельський, М. (2023). Шифрування даних в операційних системах. Матеріали VI Міжнародної студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання “, 183-184.
7. Symmetric-Key Encryption URL: https://textbook.cs161.org/crypto/symmetric.html (дата звернення: 23.05.2024).
8. MD5 vs. SHA Algorithms URL: https://www.baeldung.com/cs/md5-vs-sha-algorithms#1-security-1 (дата звернення: 23.05.2024).

9. Букатка, С., & Тимощук, В. (2023). ХЕШ-алгоритм шифрування паролів користувачів ос Linux. Матеріали VI Міжнародної студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання “, 112-113.
10. Oracle Linux URL: https://docs.oracle.com/en/operating-systems/oracle-linux/index.html (дата звернення: 23.05.2024).
11. Encrypt Drives using LUKS on Oracle Linux URL: https://docs.oracle.com/en/learn/ol-luks/index.html#introduction (дата звернення: 23.05.2024).
12. dm-crypt: Linux kernel device-mapper crypto target URL: https://gitlab.com/cryptsetup/cryptsetup/-/wikis/DMCrypt (дата звернення: 23.05.2024).
13. LUKS2 On-Disk Format Specification URL: https://gitlab.com/cryptsetup/LUKS2-docs/-/blob/main/luks2_doc_wip.pdf?ref_type=heads (дата звернення: 23.05.2024).
14. DES vs 3DES vs Blowfish vs AES URL: https://www.baeldung.com/cs/des-vs-3des-vs-blowfish-vs-aes (дата звернення: 23.05.2024).
15. Encfs - mounts or creates an encrypted virtual filesystem URL: https://linux.die.net/man/1/encfs (дата звернення: 23.05.2024).
16. FUSE URL: https://www.kernel.org/doc/html/next/filesystems/fuse.html (дата звернення: 23.05.2024).
17. cryptsetup - manage plain dm-crypt and LUKS encrypted volumes URL: https://manpages.ubuntu.com/manpages/bionic/man8/cryptsetup.8.html (дата звернення: 23.05.2024).
18. EncFSMP FAQ URL: https://encfsmp.sourceforge.io/faq.html (дата звернення: 23.05.2024).
19. Про охорону праці. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 21.05.2024).

20. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/z0356-22#n769 (дата звернення: 21.05.2024).
21. Желібо Е.Н. Безпека життєдіяльності: Навчальний посібник. Київ: Каравела, Львів: Новий світ - 2000. 2001. 320с.

ДОДАТКИ

Додаток А Процес виконання команди ініціалізації LUKS на USB накопичувачі

```
[root@oracle9u4 ~]# cryptsetup luksFormat --pbkdf pbkdf2  --hash ripemd160 --key-size 256 --cipher serpent-xts-plain64 /dev/sda --debug
# cryptsetup 2.6.0 processing "cryptsetup luksFormat --pbkdf pbkdf2 --hash ripemd160 --key-size 256 --cipher serpent-xts-plain64 /dev/sda --debug"
# Verifying parameters for command luksFormat.
# Running command luksFormat.
# Installing SIGINT/SIGTERM handler.
# Unblocking interruption on signal.
# Allocating context for crypt device /dev/sda.
# Trying to open and read device /dev/sda with direct-io.
# Initialising device-mapper backend library.
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
# Interactive passphrase entry requested.
Enter passphrase for /dev/sda:
Verify passphrase:
# Checking new password using default pwquality settings.
# Crypto backend (OpenSSL 3.0.7 1 Nov 2022 [default][legacy]) initialized in cryptsetup library version 2.6.0.
# Detected kernel Linux 5.15.0-205.149.5.1.el9uek.x86_64 x86_64.
# PBKDF pbkdf2-ripemd160, time_ms 2000 (iterations 0).
Existing 'crypto_LUKS' superblock signature on device /dev/sda will be wiped.
Existing 'crypto_LUKS' superblock signature on device /dev/sda will be wiped.
```

```
# Formatting device /dev/sda as type LUKS2.
# Auto-detected optimal encryption sector size for device /dev/sda
is 512 bytes.
# Topology: IO (512/0), offset = 0; Required alignment is 1048576
bytes.
# Checking if cipher serpent-xts-plain64 is usable.
# Using userspace crypto wrapper to access keyslot area.
# Formatting LUKS2 with JSON metadata area 12288 bytes and keyslots
area 16744448 bytes.
# Creating new digest 0 (pbkdf2).
# Setting PBKDF2 type key digest 0.
# Running pbkdf2(ripemd160) benchmark.
# PBKDF benchmark: memory cost = 0, iterations = 512000, threads =
0 (took 64 ms)
# PBKDF benchmark: memory cost = 0, iterations = 525338, threads =
0 (took 499 ms)
# PBKDF benchmark: memory cost = 0, iterations = 566185, threads =
0 (took 926 ms)
# Benchmark returns pbkdf2(ripemd160) 566185 iterations, 0 memory,
0 threads (for 256-bits key).
# Segment 0 assigned to digest 0.
# Device size 4043308544, offset 16777216.
# Wiping LUKS areas (0x000000 - 0x1000000) with zeroes.
# Wiping keyslots area (0x008000 - 0x1000000) with random data.
# Reusing open rw fd on device /dev/sda
# Device size 4043308544, offset 16777216.
# Acquiring write lock for device /dev/sda.
# Opening lock resource file /run/cryptsetup/L_8:0
# Verifying lock handle for /dev/sda.
# Device /dev/sda WRITE lock taken.
# Trying to write LUKS2 header (16384 bytes) at offset 0.
# Reusing open rw fd on device /dev/sda
#
Checksum:f3ec7e6a92aedf7efb5c7ef24c37ed54d59530750bb4a36cb21e357f4
3f33168 (in-memory)
# Trying to write LUKS2 header (16384 bytes) at offset 16384.
# Reusing open rw fd on device /dev/sda
```

```
#
Checksum:d1ab8177c7b52758c2a48baa8b588735891899307b935454bffbf99b3
768c0c0 (in-memory)
# Device /dev/sda WRITE lock released.
# Adding new keyslot -1 by passphrase, volume key provided by key
(-1).
# Selected keyslot 0.
# Keyslot 0 assigned to digest 0.
# Trying to allocate LUKS2 keyslot 0.
# Found area 32768 -> 163840
# Running pbkdf2(ripemd160) benchmark.
# PBKDF benchmark: memory cost = 0, iterations = 474898, threads =
0 (took 69 ms)
# PBKDF benchmark: memory cost = 0, iterations = 511001, threads =
0 (took 513 ms)
# Benchmark returns pbkdf2(ripemd160) 511001 iterations, 0 memory,
0 threads (for 256-bits key).
# Calculating attributes for LUKS2 keyslot 0.
# Acquiring write lock for device /dev/sda.
# Opening lock resource file /run/cryptsetup/L_8:0
# Verifying lock handle for /dev/sda.
# Device /dev/sda WRITE lock taken.
# Checking context sequence id matches value stored on disk.
# Reusing open ro fd on device /dev/sda
# Running keyslot key derivation.
# Updating keyslot area [0x8000].
# Reusing open rw fd on device /dev/sda
# Device size 4043308544, offset 16777216.
# Device /dev/sda WRITE lock already held.
# Trying to write LUKS2 header (16384 bytes) at offset 0.
# Reusing open rw fd on device /dev/sda
#
Checksum:02dfddd62a2076ffed7a018182d487758d1c9cd3b116e527f1178ec17
c3181b8 (in-memory)
# Trying to write LUKS2 header (16384 bytes) at offset 16384.
# Reusing open rw fd on device /dev/sda
```

```
#
Checksum:8fd1124a3cd5e97c73edab6131f8d73244b7adebd03595745d63f2982
2fd6f29 (in-memory)
# Device /dev/sda WRITE lock released.
Key slot 0 created.
# Releasing crypt device /dev/sda context.
# Releasing device-mapper backend.
# Closing read only fd for /dev/sda.
# Closing read write fd for /dev/sda.
Command successful.
[root@oracle9u4 ~]#
```

Додаток Б Процес створення зашифрованої файлової системи EncFS

```
[root@oracle9u4 encrypted]# encfs /mnt/usb/encrypted2
/mnt/usb/decrypted2
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: 3

Selected algorithm "CAMELLIA"

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

Select a block size in bytes.  The cipher you have chosen
```

```
supports sizes from 64 to 4096 bytes in increments of 16.
Or just hit enter for the default (1024 bytes)

filesystem block size: 4096

Using filesystem block size of 4096 bytes

The following filename encoding algorithms are available:
1. Block : Block encoding, hides file name size somewhat
2. Block32 : Block encoding with base32 output for case-insensitive
systems
3. Null : No encryption of filenames
4. Stream : Stream encoding, keeps filenames as short as possible

Enter the number corresponding to your choice: 4

Selected algorithm "Stream""

Enable filename initialization vector chaining?
This makes filename encoding dependent on the complete path,
rather then encoding each path element individually.
[y]/n: y

Enable per-file initialization vectors?
This adds about 8 bytes per file to the storage requirements.
It should not affect performance except possibly with applications
which rely on block-aligned file io for performance.
[y]/n: y

Enable filename to IV header chaining?
This makes file data encoding dependent on the complete file path.
If a file is renamed, it will not decode sucessfully unless it
was renamed by encfs with the proper key.
If this option is enabled, then hard links will not be supported
in the filesystem.
y/[n]: y
```

```
Enable block authentication code headers
on every block in a file?  This adds about 8 bytes per block
to the storage requirements for a file, and significantly affects
performance but it also means [almost] any modifications or errors
within a block will be caught and will cause a read error.
y/[n]: y

Add random bytes to each block header?
This adds a performance penalty, but ensures that blocks
have different authentication codes.  Note that you can
have the same benefits by enabling per-file initialization
vectors, which does not come with as great of performance
penalty.
Select a number of bytes, from 0 (no random bytes) to 8: 4

Enable file-hole pass-through?
This avoids writing encrypted blocks when file holes are created.
[y]/n: y

Configuration finished.  The filesystem to be created has
the following properties:
Filesystem cipher: "ssl/camellia", version 3:0:2
Filename encoding: "nameio/stream", version 2:1:2
Key Size: 256 bits
Block Size: 4096 bytes, including 12 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.

--------------------------- WARNING ---------------------------
The external initialization-vector chaining option has been
enabled.  This option disables the use of hard links on the
filesystem. Without hard links, some programs may not work.
The programs 'mutt' and 'procmail' are known to fail.  For
more information, please see the encfs mailing list.
If you would like to choose another configuration setting,
```

```
please press CTRL-C now to abort and start over.

Now you will need to enter a password for your filesystem.
You will need to remember this password, as there is absolutely
no recovery mechanism.  However, the password can be changed
later using encfsctl.

New Encfs Password:
Verify Encfs Password:
[root@oracle9u4 encrypted]#
```