Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Криптографические методы защиты информации»

Лабораторная работа №2  
«Шифрование жесткого диска. Возможности шифрования данных в *NTFS*, *ext*4, *btrfs*»

Студенты: Клоченко И.Е.

Преподаватель: Соколов И.П.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[1. Шифрование виртуального жесткого диска в *Windows* (*NTFS*) 3](#_Toc133760128)

[2. Шифрование виртуального жесткого диска в *Linux* (*Ext*4) 9](#_Toc133760129)

[Вывод 18](#_Toc133760130)

**Цель работы:**

Получение теоретических и практических навыков работы с программными средствами шифрования данных в ОС *Windows* 7 (*NTFS*) и *Linux* (*Ext*4).

**Ход работы:**

# Шифрование виртуального жесткого диска в Windows (NTFS)

1. Производим ознакомление с теоретическим материалом и запускаем виртуальную машину *Windows* 7.
2. Создаем виртуальный жесткий диск.



Рисунок 1. Создание виртуального жесткого диска

1. Инициализируем созданный виртуальный жесткий диск. После этой процедуры статус диска «Не инициализирован» сменится на «В сети».
2. Создадим том на диске, указав букву диска, его размер и тип файловой системы. Далее переходим к шифрованию файла.

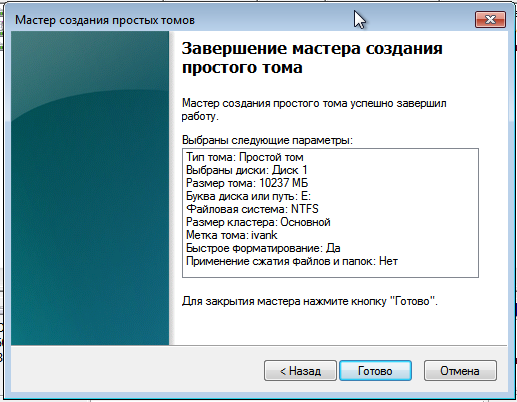


Рисунок 2. Создание простого тома на виртуальном жестком диске

1. Откроем редактор реестра (пользователь уже Администратор). По пути *HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft* *\Windows* *NT\CurrentVersion\EFS* создадим *DWORD* параметр с шестнадцатеричным значением 0x6603. Стоит отметить, что при вводе значения для получения результата нужно использовать такую запись:

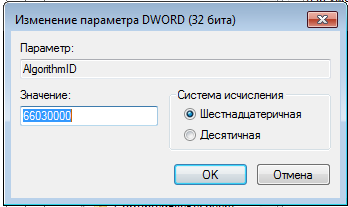


Рисунок 3. Значение (DWORD параметр AlgorithmID)



Рисунок 4. DWORD параметр AlgorithmID

1. Так как учетная запись пользователя с правами Администратора создана на этапе установки системы, новая не понадобится. Вход также уже произведен (пользователь *ivanklochenko*).
2. Создадим папку «*klochenko\_ie\_*09002025» на созданном виртуальном жестком диске.
3. Пробуем зашифровать папку «Мои документы». При попытке зашифровать эту папку, мы получаем сообщение с вопросом вида шифрования: зашифровать только эту папку или зашифровать рекурсивно все ее вложенные файлы и папки. Выбираем второе. *Windows* отображает зашифрованные папки зеленым цветом.

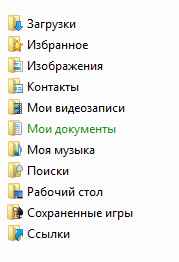


Рисунок 5. Шифрование папки «Мои документы»

1. Добавляем оснастку сертификаты для текущего пользователя через *mmc* консоль.

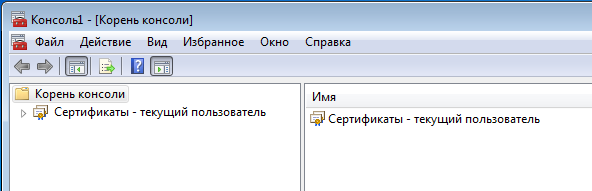


Рисунок 6. Оснастка Сертификаты – текущий пользователь

1. Просматриваем личный сертификат.

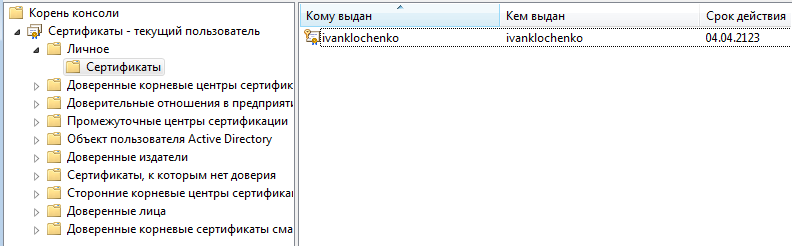


Рисунок 7. Личный сертификат пользователя

1. Экспортируем этот сертификат. Нас встречает окно «Мастер экспорта сертификатов». Следуя подсказкам графического интерфейса, выбираем «Да, экспортировать закрытый ключ», вводим пароль для закрытого ключа, указываем имя файла и место, куда будет он будет сохранен.



Рисунок 8. Экспорт секретного ключа

Переносим секретный ключ на *USB*-носитель. В моем случае файл виртуального жесткого диска находится в папке «Мои Документы», здесь же находится и папка «*keys*», в которую был сохранен файл секретного ключа. Поэтому перед тем, как перенести ключ на флешку необходимо убрать галочку «Шифровать …» в свойствах папки «Мои документы», иначе система не даст скопировать (перенести) ключ на флешку.

После переноса секретного ключа на флешку, удаляем файл на жестком диске.

1. Попробуем сымитировать случай, когда пользователь под другой учетной записью, хочет снять защиту с зашифрованных данных.

Создадим в папке *«klochenko\_ie\_09002025»* файл *some\_secret\_file*.*txt* со следующим содержимым:

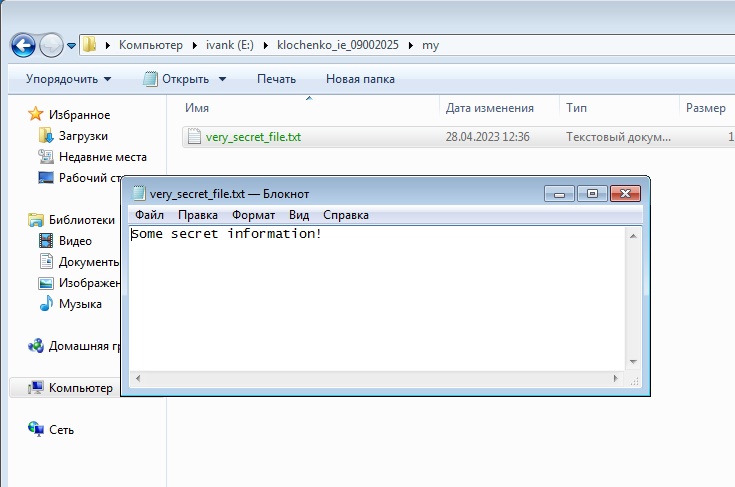


Рисунок 9. Очень секретный файл с некоторой секретной информацией пользователя ivanklochenko

Зашифруем папку:



Рисунок 10. Шифрование папки «klochenko\_ie\_09002025»

Далее активируем учетную запись администратора (была выключена по умолчанию). Понизим привилегии пользователя *ivanklochenko* до обычного пользователя. Создадим нового пользователя *attacker* с правами обычного пользователя.

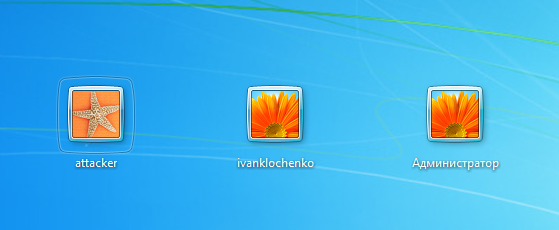


Рисунок 11. Пользователи виртуальной машины

Авторизуемся под ним. Перейдем в виртуальный жесткий диск и попробуем прочитать содержимое файла *very\_secret\_file*.*txt*:

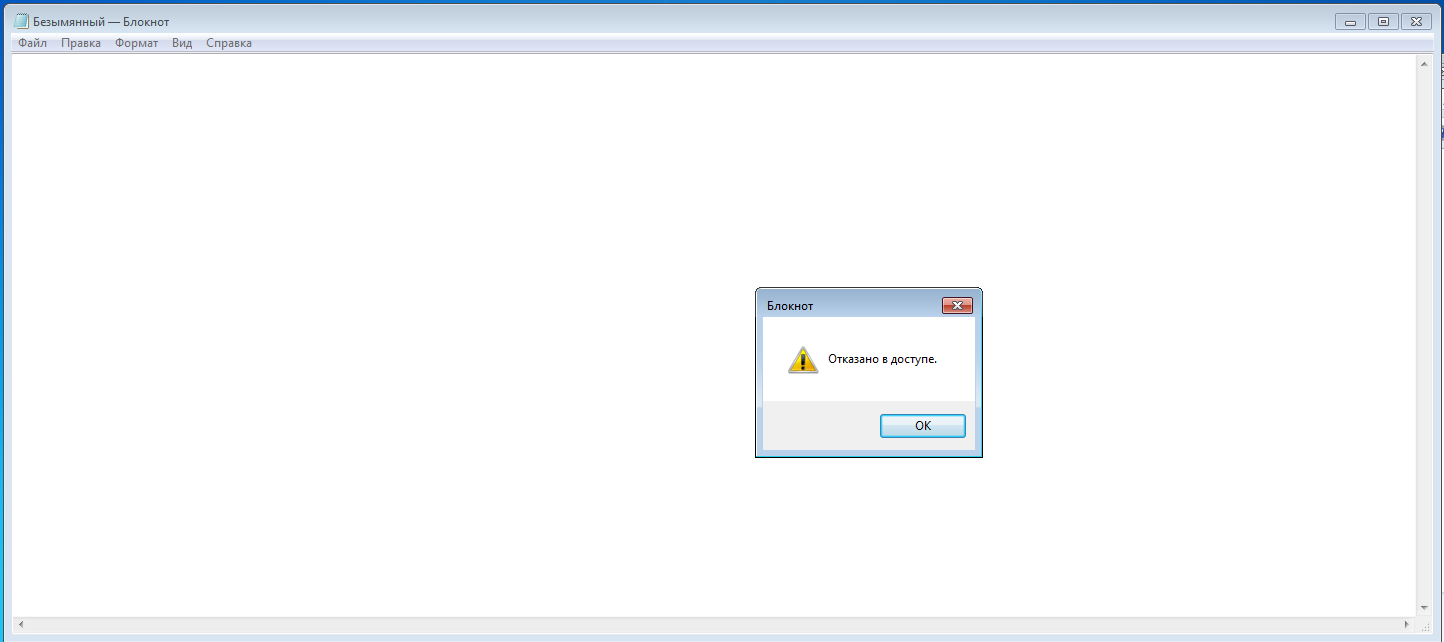


Рисунок 12. Попытка открыть зашифрованный файл пользователя ivanklochenko

Получаем ожидаемое сообщение «Отказано в доступе». Попробуем расшифровать файл, убрав галочку «Шифровать…» в свойствах файла. Получаем сообщение типа «Это может сделать только администратор». Допустим, что пользователь знает пароль от локальной учетной записи Администратор и вводим ее в предложенной форме. Но нет, даже прав администратора недостаточно, чтобы расшифровать и посмотреть содержимое зашифрованного файла, получаем следующее сообщение:

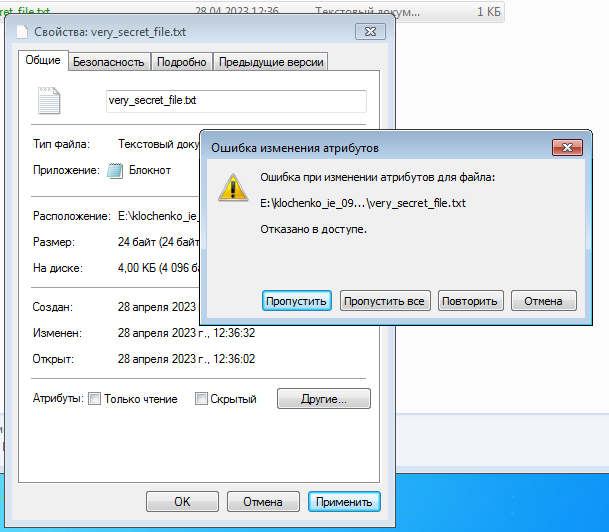


Рисунок 13. Неудачная попытка расшифровать файл

Даже после изменения владельца файла на *attacker*, открытие файла невозможно. Итог: Безопасно!

Вывод:

Шифрование в ОС *Windows* достаточно надежное, чтобы ограничить несанкционированный доступ лиц, которым не должны быть открыты шифруемые документы.

Использовать приведенный способ шифрования уместно в малых корпоративных сетях, где, например, есть один общий сетевой виртуальный диск, на котором пользователи сети хранят свои папки и файлы. Эти папки и файлы можно зашифровать, чтобы ограничить круг лиц, которые могут просматривать эти файлы. Но все-таки это несколько устаревшая методика шифрования файлов, хоть и, в некоторой степени, удобная в повседневном использовании.

# Шифрование виртуального жесткого диска в Linux (Ext4)

* 1. В качестве виртуальной машины на *Linux* используется *Kali* *Linux*.
  2. Устанавливаем *PGP*, *GPG*.
  3. Производим шифрования произвольных файлов:

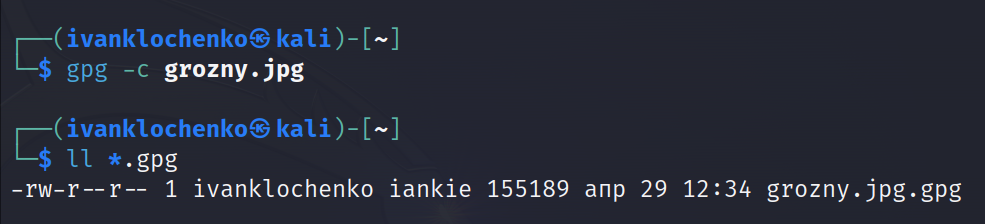


Рисунок 14. Шифрование произвольного файла

* 1. Попробуем расшифровать:

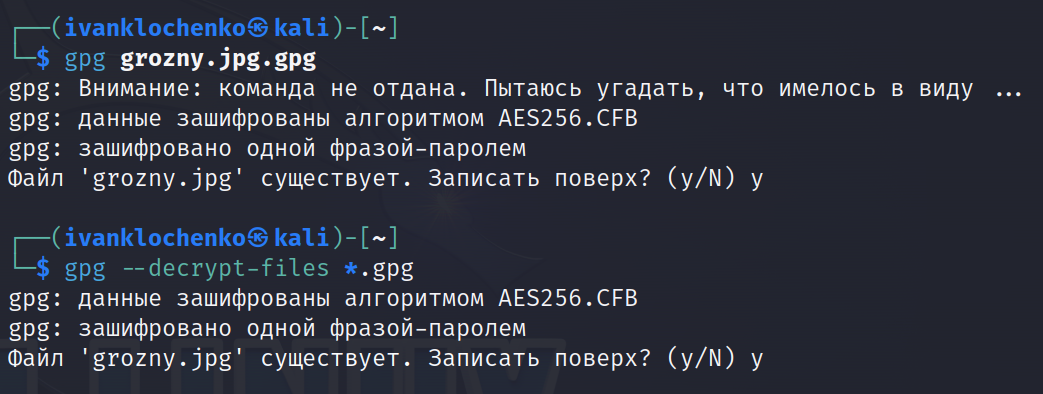


Рисунок 15. Два способа расшифровать файлы

* 1. Устанавливаем *TypeCrypt* 7.1*a* с графическим интерфейсом.

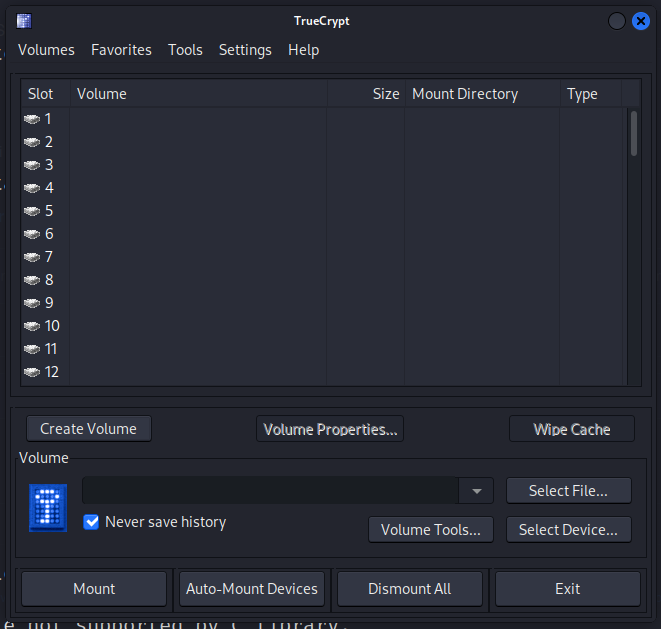


Рисунок 16. Стартовое окно Truecrypt

* 1. Создаем криптоконтейнер: *Create* *Volume* > *Create* *an* *encrypted* *file* *container* > *Standart* *TrueCrypt* *Volume*, указываем место, имя и расширение для криптоконтейнера, алгоритм шифрования (*AES* по умолчанию), далее размер создаваемого криптоконтейнера, пароль, файловая система (выбрали *Linux* *Ext*4). Далее переходим к квесту по многократному вождению мышкой для «повышения безопасности» криптоконтейнера.

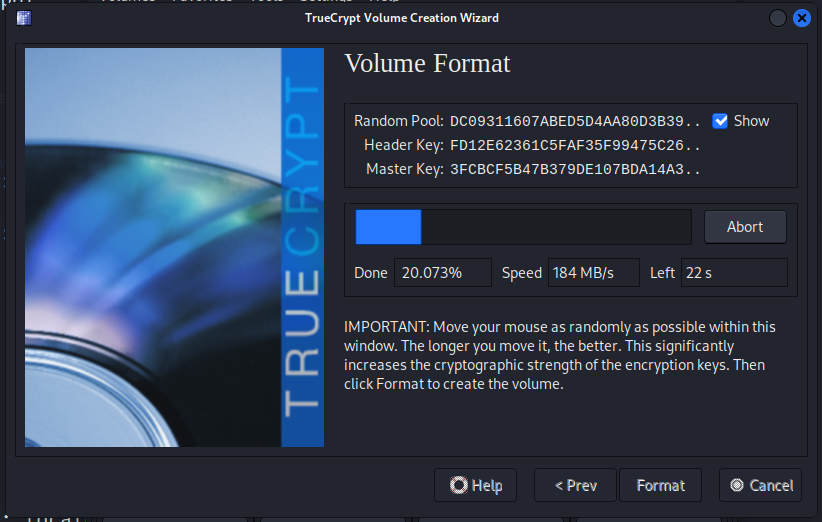


Рисунок 17. Создание криптоконтейнера в TrueCrypt

Далее программа оповещает нас о том, что криптоконтейнер создан и готов к использованию.

* 1. Примонтируем криптоконтейнер как виртуальный диск. Для этого в главном окне выбираем «*Select* *file*…» и выбираем созданный ранее файл криптоконтейнера. Вводим пароль, созданный при создании криптоконтейнера и нажимаем *OK*.

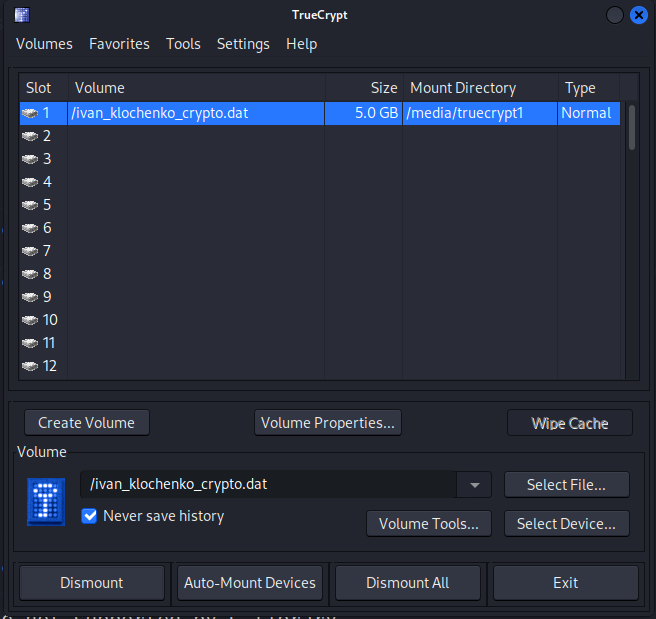


Рисунок 18. Монтирование криптоконтейнера в качестве виртуального диска

* 1. Поместим в криптоконтейнер (нынешний виртуальный диск) какой-либо файл (у меня это будет *hello*.*sh*).

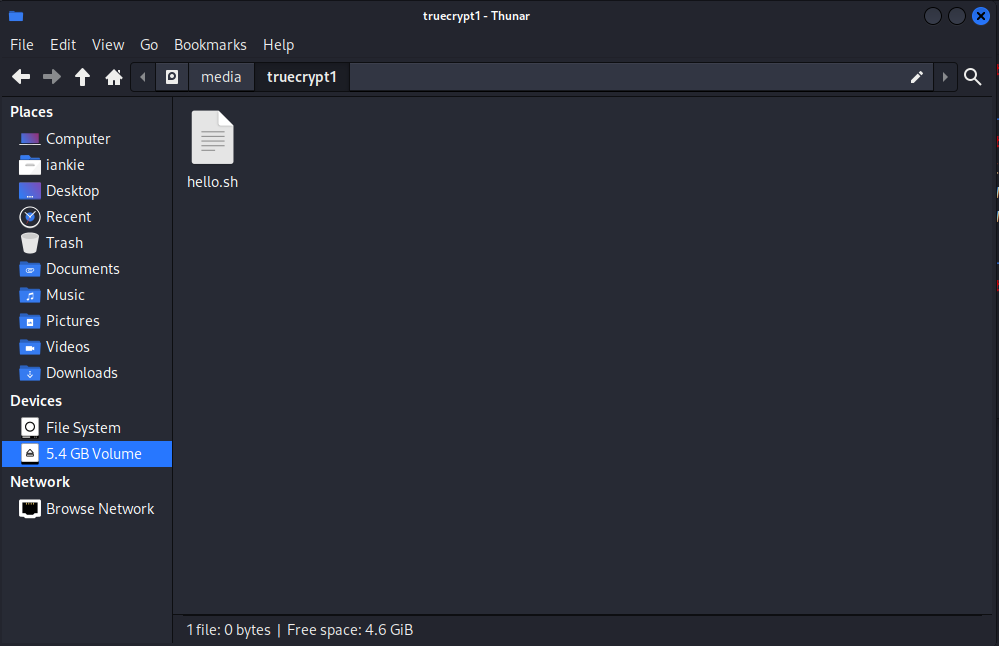


Рисунок 19. Помещение файла в криптоконтейнер

* 1. Отмонтируем диск и переместим криптоконтейнер.

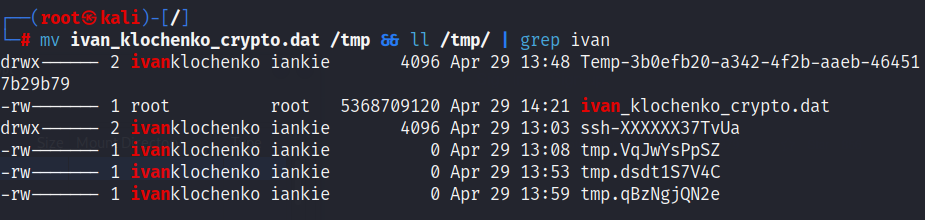


Рисунок 20. Перемещение криптоконтейнера

* 1. Снова примонтируем криптоконтейнер в качестве виртуального диска.

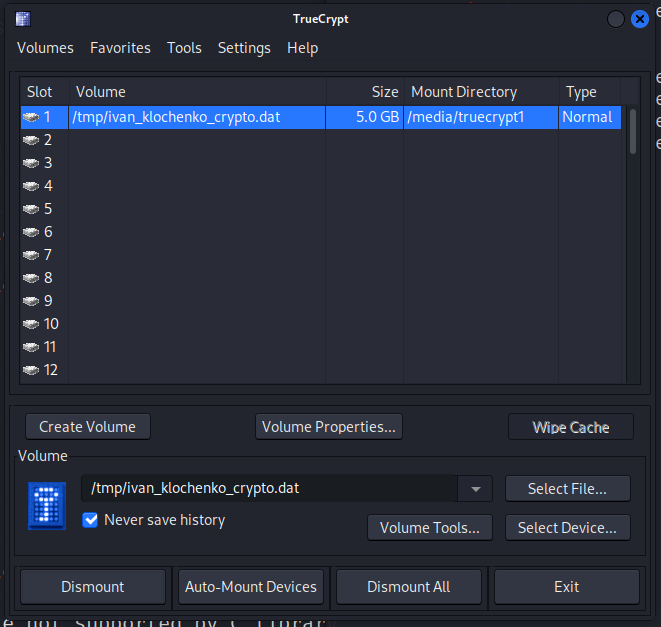


Рисунок 21. Повторное монтирование криптоконтейнера

Убедимся в том, что криптоконтейнер может передаваться и использоваться независимо, просмотрев его на наличие файла, который мы скопировали на предыдущих шагах.

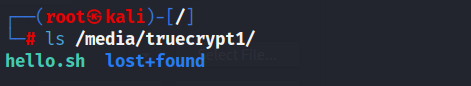


Рисунок 22. Файл в криптоконтейнере

* 1. Устанавливаем *LUKS* и *dm*-*crypt*.
  2. Создадим файл для будущего хранения зашифрованных данных - «*klochenko*». Находится этот файл в директории /*tmp*/.

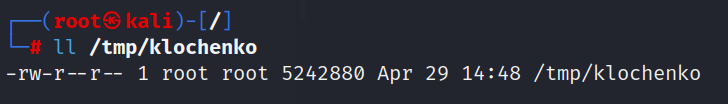


Рисунок 23. Файл для будущего хранения зашифрованных данных

* 1. Создадим криптоконтейнер, используя команду «*cryptsetup* -*y* *luksFormat* /*tmp*/*klochenko*»

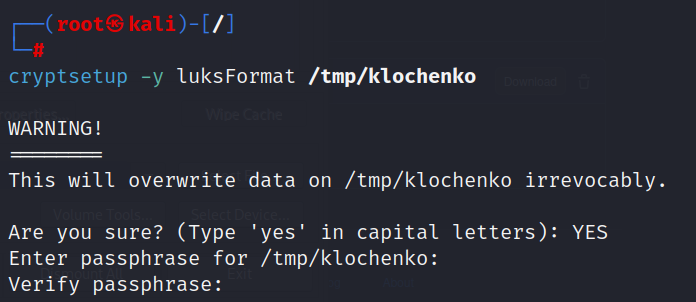


Рисунок 24. Создание кпритоконтейнера

* 1. Открываем контейнер (имя *klochenko\_cont\_*1).

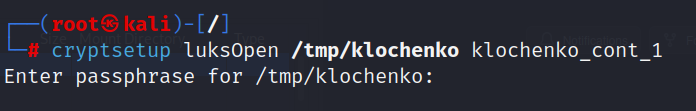


Рисунок 25. Открытие криптоконтейнера и задание ему имени

* 1. Форматируем криптоконтейнер в файловую систему *ext*4.

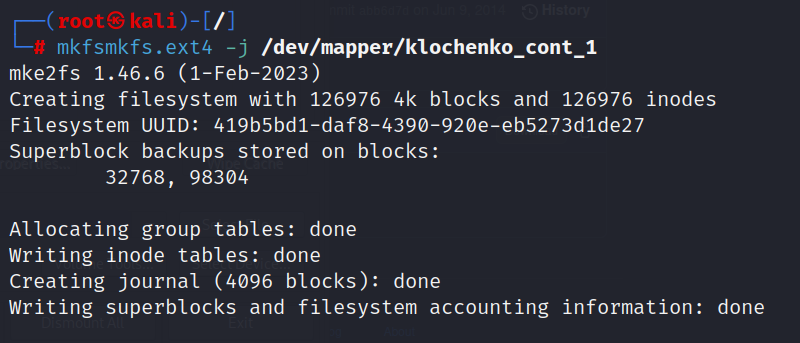


Рисунок 26. Форматирование криптоконтейнера в файловую систему ext4

* 1. Создаем папу для монтирования (также можно выражаться «точка монтирования») - /*mnt*/*klochenko\_mnt*.

Далее монтируем криптоконтейнер в качестве виртуального диска.

* 1. Скопируем папку /*etc* в криптоконтейнер.



Рисунок 27. Копирование папки /etc в криптоконтейнер

* 1. Размонтируем криптоконтейнер и закрываем криптоконтейнер.

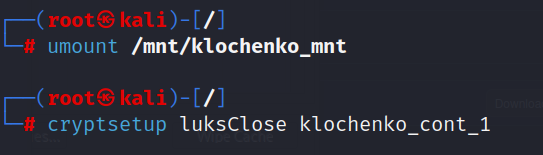


Рисунок 28. Размонтирование и закрытие криптоконтейнера klochenko\_cont\_1

Теперь наши данные зашифрованы. Для дешифровки нужно выполнить обратные действия предыдущего шага, т.е. открыть и монтировать криптоконтейнер.

Вывод:

Шифрование заняло достаточно мало времени. Чтобы ускорить шифрование можно вместо алгоритма по умолчанию *AES*-*XTS* использовать *AES*-*GCM* или *ChaCha*20-*Poly*1305 (есть вопросы по уровню безопасности этих алгоритмов). Также можно настроить *dm*-*crypt* на использование нескольких потоков для параллельного шифрования и дешифрования данных. (опция –*parallel* при создании *LUKS* тома). Еще один вариант – использовать *SSD* для хранения ключей *LUKS*.

При попытке получить доступ к файлам (здесь наверно имелось ввиду посмотреть содержимое /*mnt*/*kochenko\_mnt*) мы увидим «ничего»:

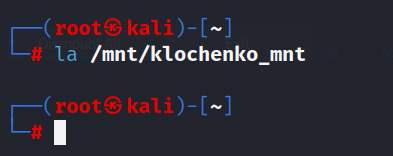


Рисунок 29. Проверка каталога /mnt/klochenko\_mnt

Результат очевиден, ведь мы демонтировали криптоконтейнер. Хорошо, теперь посмотрим на файл /*tmp*/*klochenko*:

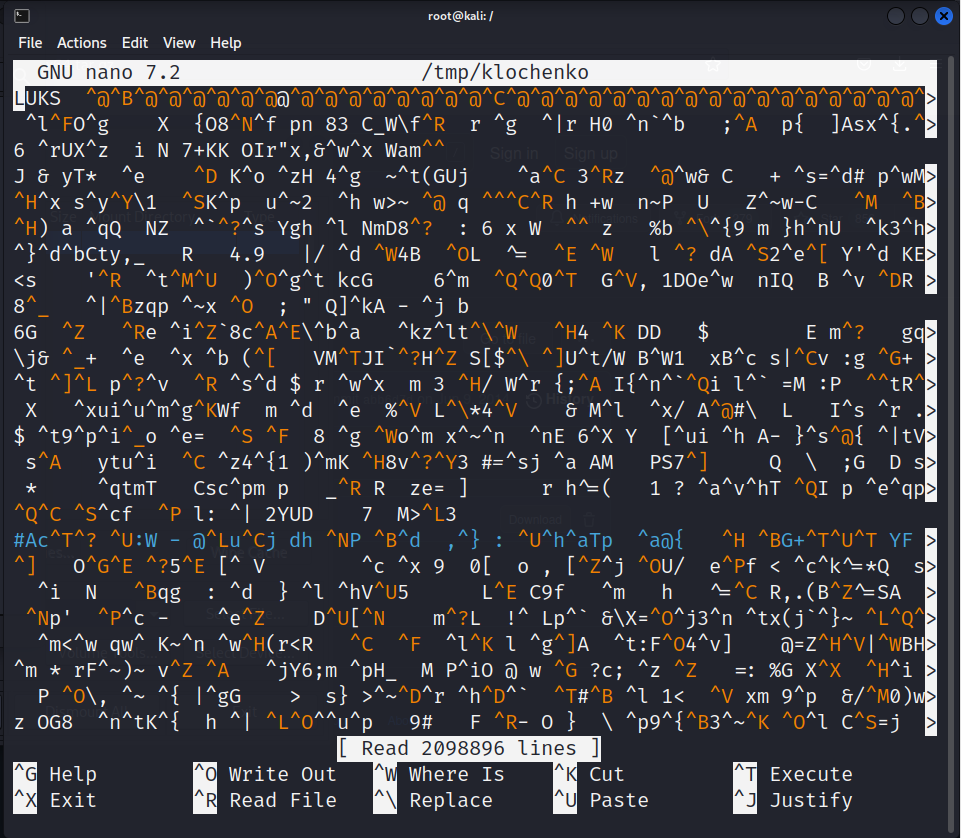


Рисунок 30. Содержимое файла /tmp/klochenko

*Magic* *bytes* показывают нам, что этот файл связан с *LUKS*. Кроме этого разобрать что-либо в этом файле невозможно.

* 1. Повторим процедуру для файловой системы *btrfs*. Для начала нужно будет установить *btrfs*-*progs*.

Создаем новый раздел.

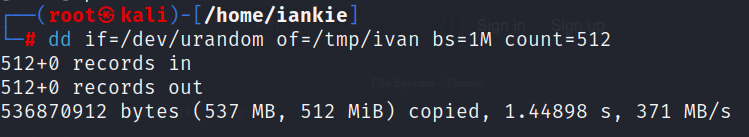


Рисунок 31. Новый раздел /tmp/ivan

Создаем криптоконтейнер.

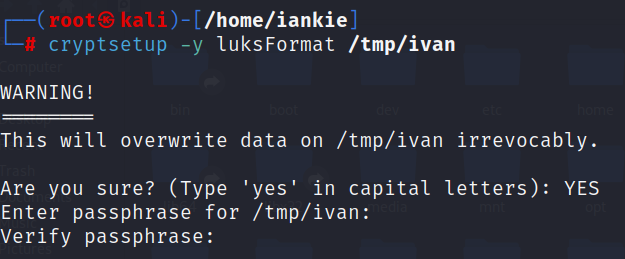


Рисунок 32. Создание криптоконтейнера для /tmp/ivan

Открываем контейнер.

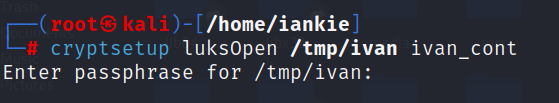


Рисунок 33. Открытие криптоконтейнера

Форматируем созданный /*dev*/*mapper*/*ivan\_cont* в *btrfs*.

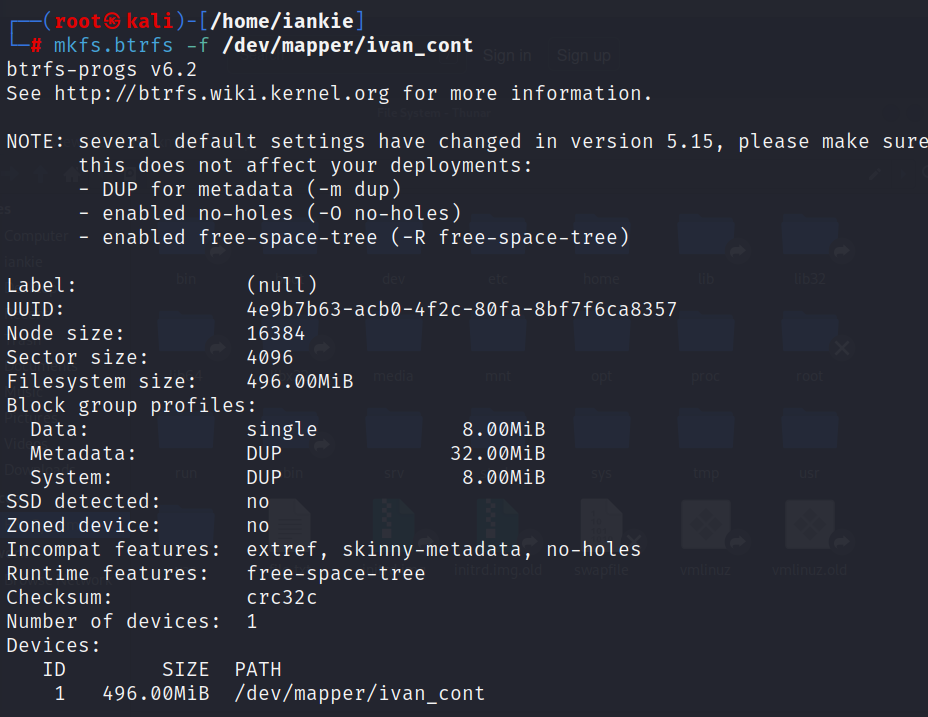


Рисунок 34. Форматирование в файловую систему btrfs

Создаем точку монтирования. Монтируем туда созданный криптоконтейнер. Копируем в криптоконтейнер любой файл (*hello*.*sh*).

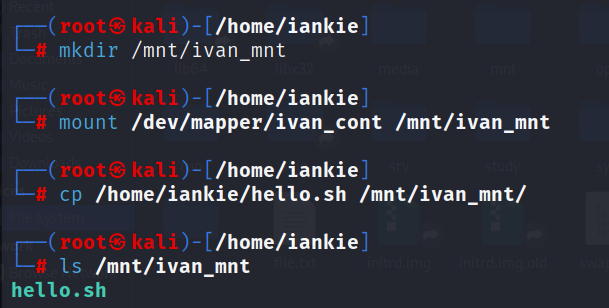


Рисунок 35. Монтирование криптоконтейнера

* 1. Попробуем переместить файлы между разделами.

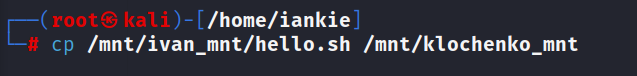


Рисунок 36. Копирование файла с раздела /mnt/ivan\_mnt на раздел /mnt/klochenko/mnt

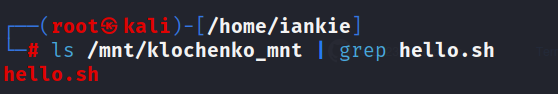


Рисунок 37. Файл hello.sh в на разделе /mnt/klochenko\_mnt

Файл успешно перенесен на другой раздел. Он также успешно открывается, исполняется и записывается. Вывод, файлы можно переносить между разделами с разными файловыми системами без потерь.

# Вывод

В ходе выполнения лабораторной работы были изучены и освоены методы шифрования жестких дисков в ОС *Windows* и *Linux* (*Kali* *Linux*).

В ОС *Windows* был рассмотрен довольно простой вариант шифрования жесткого диска, но не самый надежный на момент написания этого отчета. Использование того же *TrueCrypt* (версия для *Windows*), было бы и удобнее, и безопаснее.

В ОС *Kali* *Linux* проведена более трудоемкая работа и изучены 2 методы шифрования файлов и использованием функции монтирования и демонтирования созданных шифруемых разделов: шифрование с помощью утилиты *TrueCrypt* и шифрование через спецификацию формата шифрования дисков *LUKS* и механизм для шифрования блочных устройств *dm*-*crypt*. Принципы работы в обоих случаях схожи между собой за исключением создания ключей шифрования.

Был проведен анализ независимости содержимого контейнеров при демонтировании, переносе криптоконтейнера в другое место, и повторном монтировании в утилите *TrueCrypt*. Будет верно предположить, что такая же независимость будет наблюдаться у *LUKS* контейнеров.

Также был проведен анализ способности перемещения файлов между двум разными шифруемыми разделами с разными файловыми системами – *ext*4 и *btrfs*. Анализ завершился успешно, сделаны микровыводы.