ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «Розробка програмного забезпечення для захисту користувацьких файлів засобами мови C#»

Виконав: студент 4 курсу, групи ПД– 42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Соловйов К.В.
(прізвище та ініціали)
Керівник Золотухіна О.А.
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтроль _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____ О.В.Негоденко
«____»_____2022 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Соловйов Кирило В'ячеславович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для захисту користувацьких файлів засобами мови C#»
Керівник роботи к.т.н., доц., Золотухіна О.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "16" лютого 2022 року №22.
2. Строк подання студентом роботи 03.06.2022
3. Вихідні дані до роботи:
   3.1. Офіційна документація Microsoft.
   3.2. Visual Studio.
   3.3. Існуючі інструменти для захисту користувацьких файлів
   3.4. Наукова-технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібнорозробити):
   4.1. Аналіз та огляд існуючих додатків та інструментів для реалізації системи.
   4.2. Розробка структури додатку для шифрування користувацьких файлів.
   4.3. Програмна реалізація додатку.

4.4.Висновки.

5. Перелік графічного матеріалу

5.1 Титульний слайд
5.2 Аналоги
5.3 Мета, об’єкт, предмет та наукова новизна дослідження
5.4 Технічні завдання
5.5 Програмні засоби реалізації
5.6 Діаграма прецендентів системи
5.7 Діаграма діяльності
5.8 Екранні форми
5.9 Апробація результатів дослідження
5.10 Висновки

6. Дата видачі завдання 11.03.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми бакалаврської роботи	11.03.2022	Виконано
2	Підбір науково-технічної літератури	12.03.2022-20.03.2022	Виконано
3	Дослідження аналогів та актуальності додатку	20.03.2022-04.04.2022	Виконано
4	Аналіз та вибір інструментів для розробки додатку	04.04.2022-25.04.2022	Виконано
5	Проектування та реалізація	25.04.2022-06.05.2022	Виконано
6	Висновки, оформлення роботи	06.05.2022-13.05.2022	Виконано
7	Передзахист	16.05.2022-03.06.2022	
8	Захист роботи		

Студент _____ К.В.Соловйов
(підпис) (прізвище та ініціали)

Керівник роботи_____ О.А.Золотухіна

РЕФЕРАТ

Мета роботи: розробити додаток, що вирішить проблему незахищеності користувацьких файлів за допомогою їх шифрування криптографічним алгоритмом.

Проаналізовано наявні програми які вирішують описану проблему, описано їх переваги та недоліки та сформульовано вимоги й задачі, яким повинна відповідати програма.

Обсяг дипломної роботи складає _ сторінок , _ рисунків, 14 використаних джерел літератури.

Функціонал системи розбито на підмодулі. Змодельовано та реалізовано архітектуру взаємодії всіх підсистем в рамках цілісної системи, що дозволяє легко розширювати та масштабувати продукт. Описано можливі варіанти взаємодії користувача з програмною системою.

Ключові слова : шифрування, MVVM, C#, .NET, криптографія, Avalonia.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVVM – Model-View-VeiwModel

AES - Advanced Encryption Standard

UML - Unified Modeling Language

UI – User Interface

ПЗ – Програмне Забезпечення

XAML - eXtensible Application Markup Language

ВСТУП

Двадцять перше століття – це час неймовірного розквіту інтернету, комп’ютерів, смартфонів та інших електронних носіїв інформації. Доступ до інтернету мають 4,66 мільярди людей , а це 59,5% всього населення нашої планети, кожен з цих людей створює гігабайти інформації в день , не тільки публічної , але й конфіденційної вчасності , це можуть бути як звичайні фотографії , так і важливі документи. Всі ці фактори є сприятливим середовищем для злочинців, які хочуть оволодіти цією інформацією, для свого фінансового збагачення , попри всі моральні цінності. Для того щоб захистити свої файли використовують спеціальне програмне забезпечення , яке шифрує інформацію та не дає хакерам до неї дістатися.

Користувачі електронних носіїв інформації доволі часто не дуже безпечно користуються інтернетом , переходять по підозрілим посиланням , скачують файли з непровірених сайтів і тому подібне. Така поведінка користувачів призводить до зараження свого пристрою шкідливим програмним забезпеченням , яке має доступ до всіх фалів які є на пристрої користувача. Саме тому розробка програмного забезпечення для шифрування користувацьких фалів є актуальним в наш час.

При шифруванні файлів зловмисник має доступ не до початкового файлу , а вже до перекодованої копії. Хакер не буде знати ні що за вид файлу було закодовано , ні алгоритму кодування , все що він може зробити з даними які викраде це піти за російським кораблем.

Дана програма розроблена для всіх людей які хочуть захисти свої конфіденційні файли від третіх осіб.

Програмне забезпечення має зручний та зрозумілий інтерфейс , який не перенасичений лишньою інформацією та при цьому виконує всі свої основні функції.

На теперішній час проблема захисту користувацьких файлів має чимало варіантів вирішення. Програми мають велику кількість алгоритмів шифрування та допоміжних функцій. Деякі створюють зашифровані віртуальні диски, деякі просто шифрують всю систему загалом.

Проте , більша кількість цих програм не задовольняють проблеми потенційних користувачів. Загалом вони створення для користувачів які розуміють який вид алгоритму шифрування їм потрібен , ці програми не задовольняють користувачів які звикли скачати , відкрити програму й одразу почати працювати з нею. Інтерфейс програм для шифрування є не тільки застарілим , а ще й орієнтованим на досвідченого користувача.

Призначенням програмного забезпечення , яке розробляється є вирішення проблеми незахищеності файлів користувачів від третіх осіб.

Потенційними користувачами програмного забезпечення є люди які не мають знань в алгоритмах шифрування даних, але хочуть захистити свою конфіденційну інформацію.

1.АНАЛІЗ ПРОБЛЕМИ НЕЗАХИЩЕНОСТІ КОРИСТУВАЦЬКИХ ФАЙЛІВ ТА ІНСТРУМЕНТІВ ДЛЯ ЇХ ЗАХИСТУ

1.1 Поширення електронних носіїв інформації та зростання кількості кібератак

Зараз, в час розквіту інформаційних технологій, людей які мають персональний комп'ютер, ноутбук, телефон, чи планшет, становиться все більше. Компанії випускають нові електронні носії інформації та комплектуючі до них , майже кожен рік , через те доступність до нових технологій для звичайних людей становиться все більше. Ми не можемо уявити своє життя без них , адже вся інформація яка тобі потрібна є згрупованою в одному пристрої. Так зростання доступності технологій які забезпечують вихід до інтернету можливо глянути на графіку динаміки проникнення інтернету в Україні Рисунок 1.1.1

Рисунок 1.1.1 – Динаміка проникнення інтернету в Україні

Згідно з даними дослідження, 66% інтернет-користувачів використовують для виходу в Інтернет смартфон, 40% - домашній ноутбук, 36% - стаціонарний домашній комп’ютер, 5% - стаціонарний комп’ютер на роботі. З даними можливо ознайомитись на Рисунку 1.1.2.

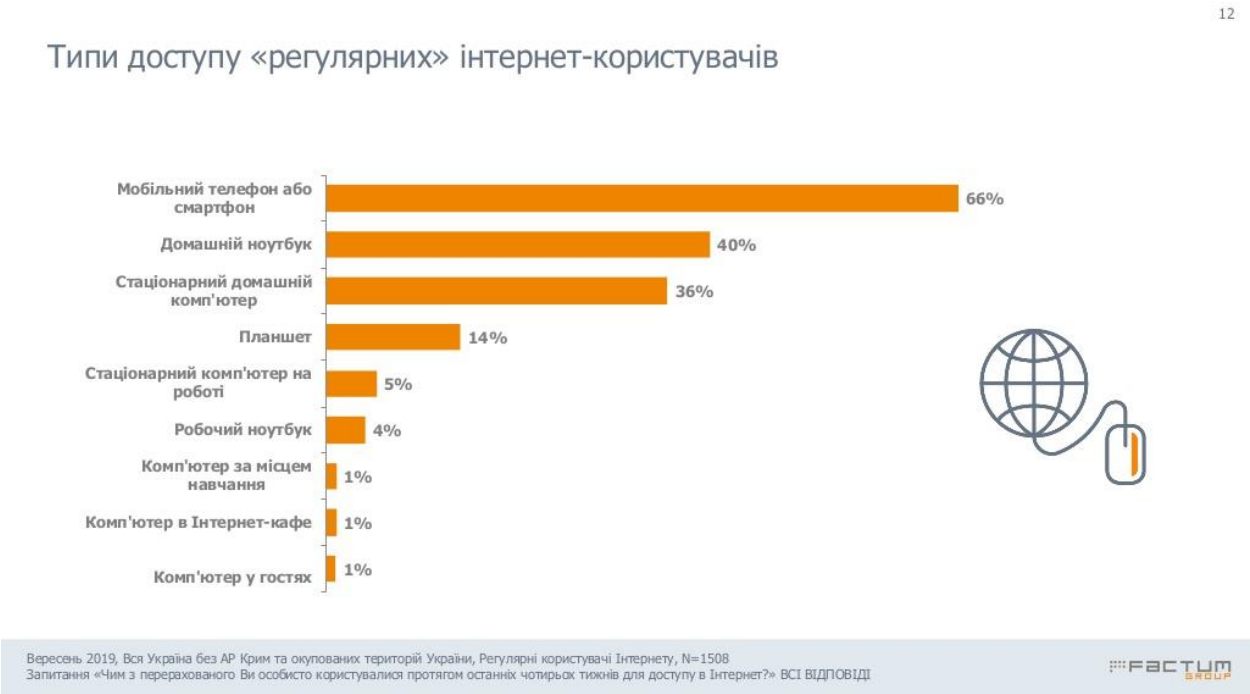

Рисунок 1.1.2 – Типи доступу до регулярних інтернет-користувачів

В слід за зростанням доступності інформаційних технологій , зростає кількість та якість кібератак. Кібератака — це шкідливе втручання в інформаційну систему компанії, злом сайтів і додатків, особистих акаунтів і пристроїв. Вони відрізняються по способу дії. Найпопулярніші види:

– Шкідливе ПЗ - вірусні програми, які заражають пристрій. Вони блокують роботу пристрою або окремих сервісів, встановлюють програми для збору даних та стеження, копіюють та знищують файли;
– Фішинг - розсилання повідомлень або електронних листів із шкідливим кодом. При переході за посиланням шахраї отримують доступ до ваших особистих та платіжних даних;

– Уразливість нульового дня - коли атакують уразливості, які самі розробники ще виявили і встигли усунути.

Так с кожним роком, через хакерські атаки, організації по всьому світу втрачають мільярди доларів. Інфографіка з цими втратами зображена на Рисунку 1.1.3.

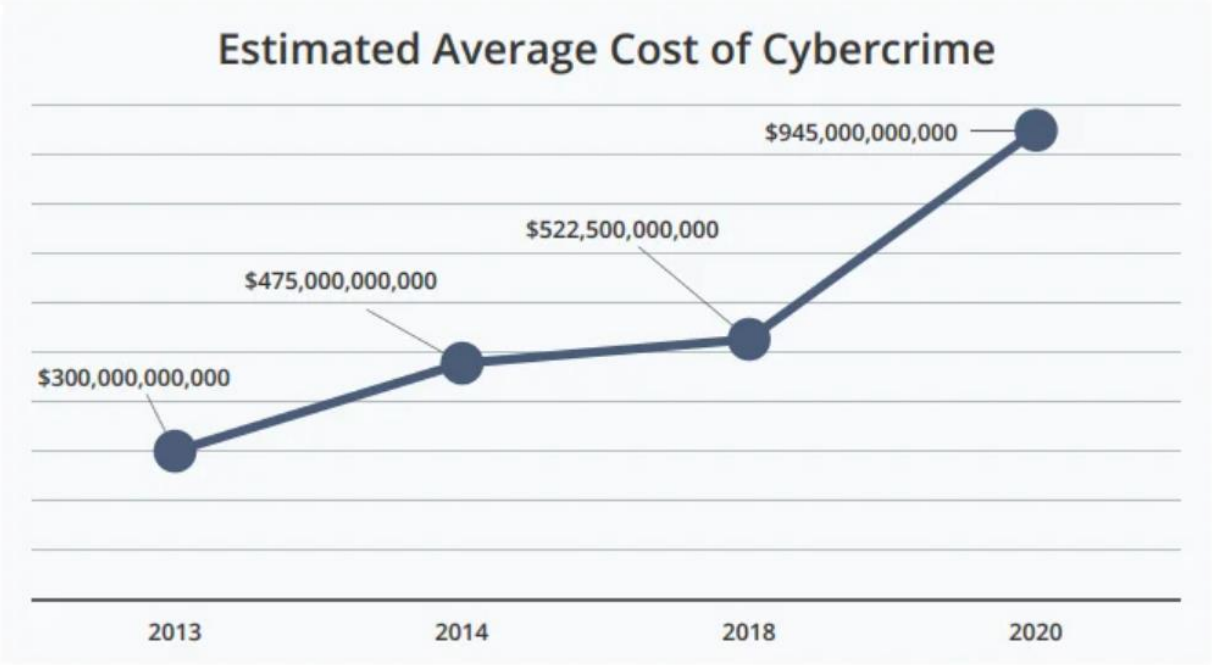

Рисунок 1.1.3 - Інфографіка: Звіт Приховані витрати кіберзлочинності / McAfee

Звичайні користувачі страдають від кіберзлочинців не менше чим великі компанії. Найвідоміші атаки на користувачів:

– Stuxnet – 2009 - 2010 роки – мережевий вірус, який вразив особисті комп'ютери, а також автоматизовані системи управління виробництвом. Він використовував чотири вразливості "нульового дня" в ОС Windows. Вірус успів заразити близько 200 тисяч пристроїв.
– WannaCry – 2017 рік – шкідлива програма-вимагач, яка використовувала вразливість нульового дня у різних версіях Windows. Проникаючи в комп'ютери, вірус зашифровував весь вміст,

а потім вимагав гроші за розблокування. Програма заразила понад 500 тисяч комп'ютерів в 150 країнах світу;

1.2 Огляд програм та інструментів для захисту користувацьких файлів

Для захисту користувацьких файлів від несанкціонованого доступу існує безліч рішень:

- Антивіруси – програмне забезпечення яке захищає пристрій від шкідливих програм. Видаляє уражені файли як тільки вони появилися на пристрої. Ефективно захищає лише від вже досліджених вірусів;
- Зовнішні накопичувачі – флешки , жорстокі диски , SSD-диски. Для збереження важливих файлів їх копіюють на зовнішні накопичувачі і зберігають без підключення до пристрою, щоб завжди мати копію важливих даних;
- Програми шифратори – використовують криптографічні алгоритми шифрування для перекодування файлу. Після використання цих додатків до файлів має доступ лише користувач в якого є ключ-шифрування;

Серед цих рішень виділяються програми шифратори , адже вони захищають файли і не дають доступ до них навіть коли пристрій вже є зараженим. Схема роботи цих додатків має деякі відмінності. Деякі програми створюють зашифрований віртуальний диск , на якому користувач може зберігати свої дані. Інші зашифровують сам файл , при цьому запропоновують користувачу вибрати алгоритм шифрування.

Рисунок 1.2.1 – Візуалізація роботи симетричних програм шифрування користувацьких даних

Важливою частиною програми є сам алгоритм шифрування, який схожий на чорний ящик. Документ, відео , зображення або інший файл, який ви завантажуєте в нього, ви отримуєте назад. Але те, що ви бачите, здається маячнею. Для цього використовуються різні симетричні та асиметричні алгоритми. На Рисунках 1.2.1 та 1.2.2 зображена робота симетричного та асиметричного алгоритмів шифрування

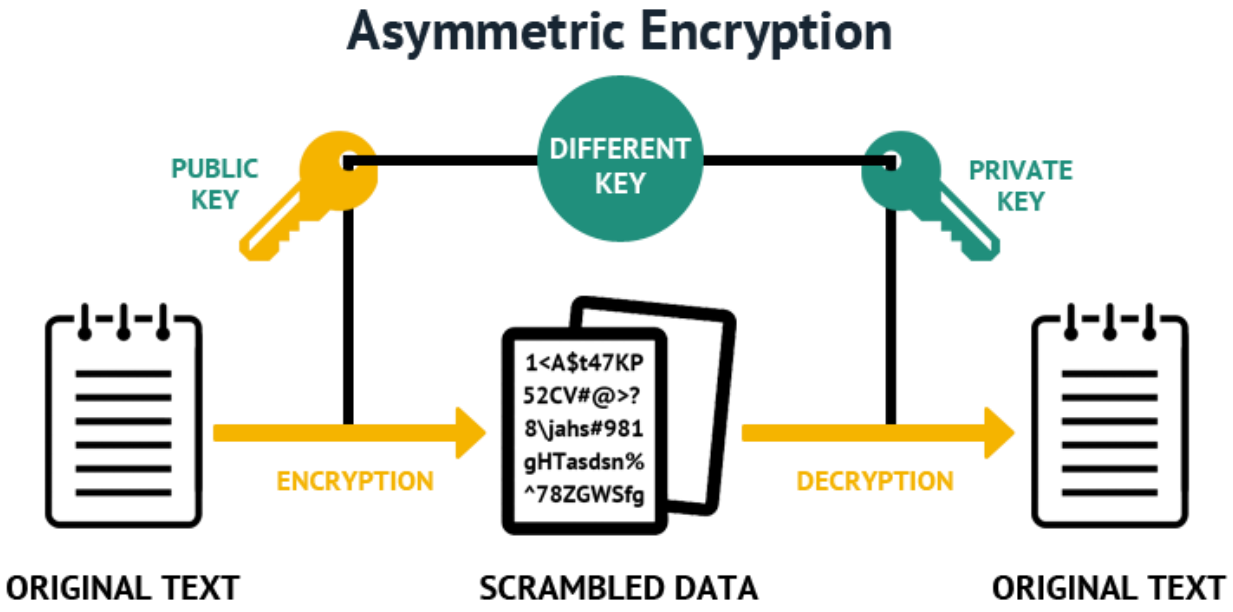

Рисунок 1.2.2 – Візуалізація роботи асиметричних програм шифрування користувацьких файлів

Самі програми дають вам вибір алгоритму, щоб ви могли використати саме той який вам потрібен виходячи з поставлених задач. Приклади алгоритмів :

– AES (advanced encryption system);

– TDEA (triple data encryption algorithm) ;

– DES (data encryption standard);

– RSA (Ron Rivest, Adi Shamir, Leonard Adleman звідси «RSA»);

Приклади програм шифрування користувацьких файлів:

1. TrueCrypt
2. CipherShed
3. BitLocker

1.2.1 TrueCrypt

TrueCrypt – це легендарна програма для шифрування користувацьких даних. Була розроблена ентузіастами ще в 2004-му році, на той час майже не існувало аналогів і тому проект швидко завоював ринок и став батьком для всіх інших подібних програм. Як виглядає інтерфейс користувача можливо побачити на Рисунку 1.2.1.

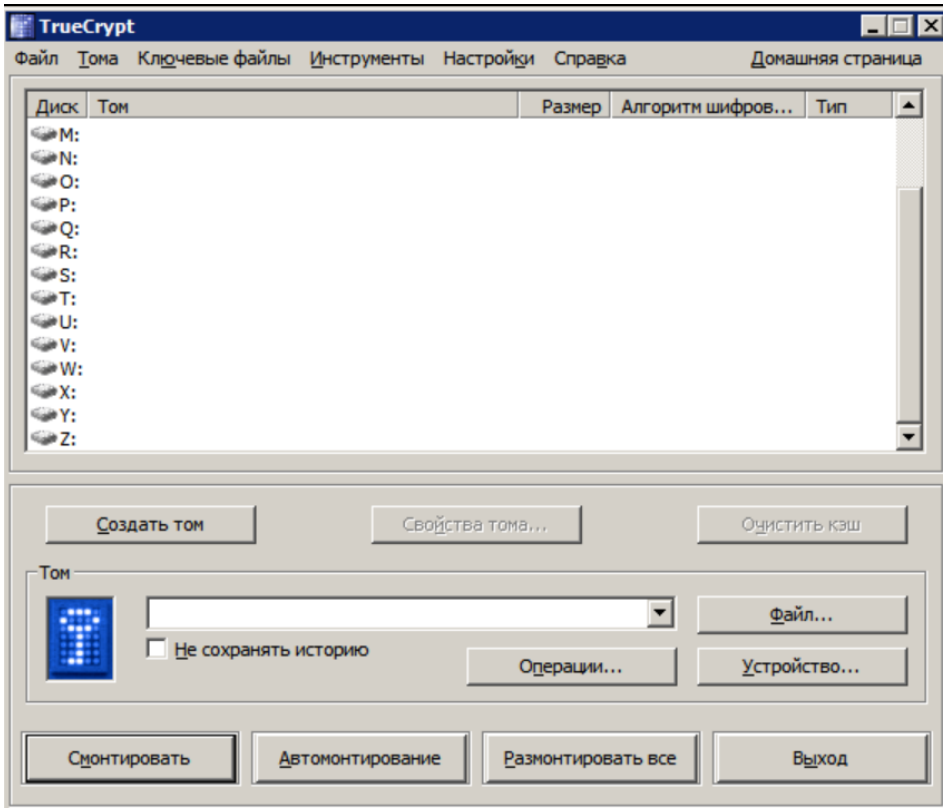

Рисунок 1.2.1 – Інтерфейс програми TrueCrypt

Розробка цього додатку була завершена в 2012 році, офіційно проект був закритий 28 травня в 2014 році, а сам розробник тихо пішов займатися іншим додатком.

Історія програми в 2012 році не зупинилась , так для перевірки безпеки в 2013 році розпочався збір коштів для проведення незалежного аудиту TrueCrypt. Причиною стала отримана від Едварда Сноуден інформація, про навмисне ослаблення засобів шифрування TrueCrypt. На аудит було зібрано понад 60 тисяч доларів. На початку квітня 2015 року роботи були завершені, але жодних серйозних помилок, чи вразливостей та інших суттєвих недоліків в архітектурі програми виявлено не було, але вже в вересні цього року було виявлено дві критичні вразливості (CVE-2015-7358, CVE-2015-7359) . Одразу після аудиту фахівці компанії ESET опублікували звіт про те, що російськомовна версія TrueCrypt 7.1a , завантажена з сайту truecrypt.ru, містила малвар. Більш того, сам сайт truecrypt.ru використовувався як командний центр - з нього відправлялися команди інфікованих комп'ютерів.

Функціональність додатку:

- Створення шифрованого віртуального диску в фалі-контейнері
- Створення шифрованого віртуального диску в виді зашифрованого розділу диску
- Створення шифрованого віртуального диску шляхом шифрування вмісту пристрою
- Створення зашифрованого динамічного файлу на дисках NTFS
- Шифрування даних одним с трьох алгоритмів : AES, Serpent , Twofish.
- Генерація ключа з вибором хеш-функції: HMAC-RIPEMD-160, HMAC-Whirlpool, HMAC-SHA-512.

Переваги:

1. Безкоштовний доступ до всіх функцій додатку.
2. Додаток можливо запускати без установки в операційній системі.
3. Можливість назначати комбінації клавіш для шифрування та дешифрування томів.
4. TrueCrypt дає користувачу вибір алгоритму шифрування та вибір хеш-функцій для генерації ключів шифрування.
5. Різновид створення шифрованих віртуальних дисків дає користувачу змогу вибрати найзручніший для нього.
6. Підтримка тридцяти язиків інтерфейсу.

Недоліки:

1. Так як більшість операційних систем встановлені на GPT-диски, то перед початком роботи з TrueCript, GPT-диск потрібно форматувати в формат MBR.
2. Підтримка проекту була завершена в 2014 році , тому нових версій більше не буде.

3. Велика популярність програми породила її злоякісну копію , тому є вірогідність натрапити на троян.
4. Технології які використовує програма для роботи з файлами є застарілими.
5. Оновлення додатку випускались до 2012 року , але інтерфейс майже не змінювався за 2004-го.
6. В 2015 році було виявлено дві критичні вразливості (CVE-2015-7358, CVE-2015-7359)

1.2.2 Bitlocker

BitLocker Drive Encryption – це пропрієтарна технологія, яка є частиною операційних систем Microsoft Windows Vista, Windows 7, Windows 8, Windows Server 2008 R2, Windows 10 та Windows 11. На Рисунку 1.2.2 зображено логотип цього додатку.

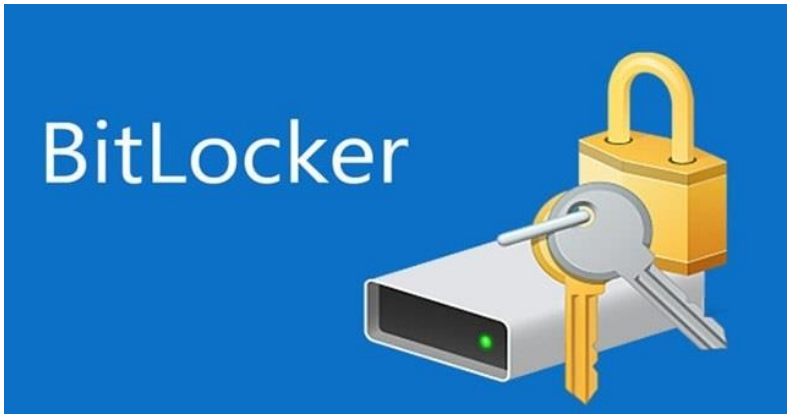

Рисунок 1.2.2 – Логотип додатку Bitlocker

BitLocker шифрує том, а не фізичний диск. Том може займати частину диска, а може включати масив з декількох дисків. Для роботи BitLocker у разі шифрування системного диска потрібно два NTFS-томи, один для ОС і один для завантажувальної частини. Останній повинен бути не менше 1,5 Гб і не буде зашифрований. Починаючи з Windows Vista SP1, з'явилася можливість шифрувати несистемні томи. Після створення розділів необхідно ініціалізувати TPM-модуль на ПК, де він є, та активувати BitLocker. У

Windows 7 з'явився BitLocker To Go, що дозволяє шифрувати змінні носії, а також знижено вимоги для завантажувальної частини, для неї достатньо 100 Мб. Якщо Windows 7 інсталюється на порожній диск, завантажувальний розділ створюється автоматично.

Ця програма підтримує наступні алгоритми шифрування : AES 128, AES 128 Elephant diffuser, AES 256 , AES 256 Elephant diffuser.

Сам ключ може зберігатися в TPM або USB-пристрої, або ж на комп'ютері. У випадку з TPM під час завантаження комп'ютера ключ може бути отриманий з нього відразу, або тільки після автентифікації за допомогою USB-ключа або введення PIN-коду користувачем.

Переваги:

1. Bitlocker є частиною операційної системи Windows, тому не потребує скачування та установки.
2. Є можливість працювати з GPT та MBR-дисками.
3. Вибір алгоритмів є невеликим і користувачу легко вибрати який йому потрібно. Тобто користувач може вибрати швидке шифрування AES 128 яке не навантажує систему , але дає невеликий захист файлів в порівнянні з AES-128 с Elephant diffuser, який є більш вимогливим до системи.
4. Об’ємна кількість комбінацій доступу до зашифрованих файлів: TMP, TMP + PIN, TMP + PIN + USB-ключ, TMP + USB-ключ, USB-ключ, PIN.
5. Програма підтримує шифрування USB-пристроїв.

Недоліки:

1. Bitlocker працює тільки за операційною системою Windows.
2. Закритий код програми не дає можливості перевірити додаток на наявність в ній бекдорів, які розробники могли спеціально залишити для спецслужб.

3. Прозорий режим роботи програми є уразливим для так званої холодної загрузки.
4. Режим USB-ключа є уразливим до буткіт-нападу.
5. Bitlocker не може шифрувати системний том, на якому встановлена операційна система.

1.2.3 CipherShed

CipherShed - безкоштовна комп’ютерна програма, яку розробив засновник TrueCrypt. Створена для шифрування файлів або цілих дисків (включаючи USB-накопичувачі та зовнішні жорсткі диски). Цей додаток має інтерфейс , за допомогою якого , користувач може повноцінно користуватись програмою. Після створення зашифрованого файлу або диска, зашифрований том монтується через CipherShed. Під'єднаний том відображається як звичайний диск, який можна читати та записувати на льоту. Шифрування прозоре для операційної системи та будь-яких програм. На Рисунку 1.2.3 зображено інтерфейс цього додатку.

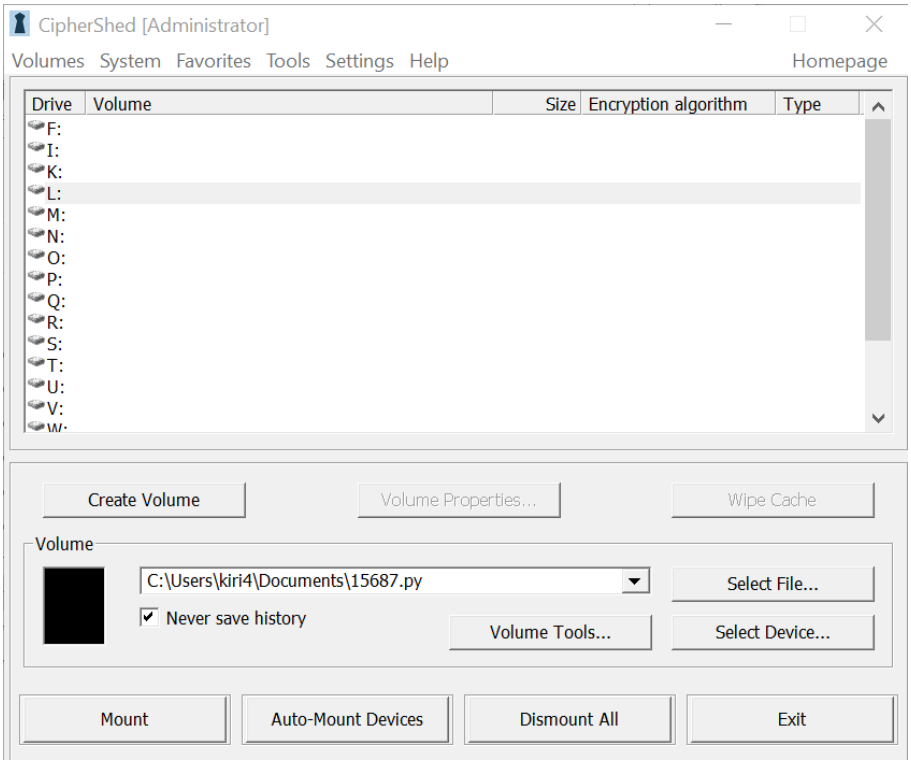

Рисунок 1.2.3 – Інтерфейс додатку CipherShed

Функціональність додатку:

- Шифрування файлів
- Створення шифрованих дисків
- Шифрування USB-накопичувачів

Переваги:

1. CipherShed – є безкоштовною програмою, тому при скачування додатку користувачу буде доступний повний функціонал шифрувальника.
2. Цей проект має відкритий код , чим показує свою відкритість до людей и говорить, про те що йому немає чого приховувати.
3. Так як CipherShed є наслідником TrueCrypt користувач може працювати с даними які були зашифровані батьківською програмою.
4. Кросплатформість проекту позволяє його запускати на різних операційних системах, а саме: Windows, Linux. Mac.
5. Є можливість вибрати варіант створення зашифрованого віртуального диску: в виді файл-контейнеру , зашифрованого розділу диску , та повне шифрування вмісту пристрою.

Недоліки:

1. CipherShed не здатний працювати з GPT-дисками.
2. Не зважаючи на те що підтримка додатку все ще якось існує, сама програма вже є застарілою, так як основою для його роботи є проект TrueCrypt(2004р).
3. UI-інтерфейс не відповідає сучасній концепції інтерфейсу. В даній програмі він має основу інтерфейсу файлового менеджеру Windows XP, що може відштовхувати нових користувачів.
4. Нові версії програми виходять надзвичайно довго на стільки , що про нову версію ходять тільки слухи, але автори від проекту не відмовляються , тому користувачі можуть лише надіятися та чекати.

1.3 Засоби розробки системи

1.3.1 Мова програмування C#

C# — це об'єктно-орієнтована мова програмування. Розроблена групою інженерів Microsoft у 1998-2001 роках як мова програмування для Microsoft .NET Framework та .NET Core.

C# був розроблений як мова програмування прикладного рівня для CLR, тому він переважно залежить від можливостей самого CLR. Здебільшого це стосується системи типів C#, яка надає BCL. Наявність чи відсутність певних виразних ознак мови залежить від того, чи можна перекласти конкретні мовні особливості у відповідні конструкції CLR. Отже, з розвитком CLR від версії 1.1 до версії 2.0, сам C# став набагато багатшим.

Синтаксис мови усуває складність C++ і надає потужні функції, такі як нетиповані значення, перерахування, делегати, лямбда-вирази та прямий доступ до пам’яті. C# підтримує універсальні методи та типи, які підвищують безпеку типів та продуктивність. Ітератори дозволяють класам творців колекцій визначати кастомні поведінки ітерацій, які легко використовувати у коді клієнта. Вирази інтегрованої мови запитів (LINQ) роблять строго типізований запит першокласною конструкцією мови.

C# — це об’єктно-орієнтована мова, що означає, що вона підтримує інкапсуляцію, успадкування та поліморфізм. Клас може успадковувати безпосередньо від одного батьківського класу, але може реалізувати будь-яку кількість інтерфейсів. Методи, які замінюють віртуальні методи в батьківських класах, повинні блокувати ключове слово override, щоб запобігти випадковому перевизначенню. На додаток до всіх переваг ООП, мова C# полегшує розробку програмних компонентів за допомогою кількох інноваційних мовних конструкцій:

- Інкапсульовані сигнатури методів, названі делегатами, включають оповіщення безпеки типів:
- Властивості служать акцесорами до змінних закритих елементів.
- Атрибути надають декларативні метадані щодо типів під час виконання.
- Рядкові документаційні коментарі XML.
- Інтегрована мова запитів (LINQ) надає вбудовані можливості запитів між джерелами даних.

Так як C# має підтримку .NET Core , та є C-подібною мовою, підтримую дуже велику кількість бібліотек , він гарно підходить для рішення проблеми незахищеності користувацьких файлів.

1.3.2 Алгоритм шифрування AES

AES – це симетричний алгоритм блочного шифрування, з розміром блоку 128-біт та ключу 128/198/256-біт, 26 травня 2002 року був прийнятий в якості стандарту шифрування урядом США. Алгоритм є добре проаналізованим і має широкий спектр використання.

AES є стандартом, що базується на алгоритмі Rijndael. Для AES довжина input (блоку вхідних даних) і State (стану) стала і дорівнює 128 біт, а довжина ключа шифрування K становить 128, 192, або 256 біт. При цьому вихідний алгоритм Rijndael допускає довжину ключа та розмір блоку від 128 до 256 біт з кроком 32 біти. Для позначення вибраних довжин input, State та Cipher Key у 32-бітних словах використовується нотація Nb = 4 для input та State, Nk = 4, 6, 8 для Cipher Key відповідно для різних довжин ключів. На початку зашифровування input копіюється до масиву State за правилом:

$state[r, c] = input[r + 4c]$, для $0 \leq r < 4$ і $0 \leq c < Nb$

Після цього State застосовується процедура AddRoundKey(), і потім State проходить через процедуру трансформації (раунд) 10, 12, або 14 разів (залежно від довжини ключа), при цьому треба врахувати, що останній раунд трохи

відрізняється від попередніх. У результаті, після завершення останнього раунду трансформації, State копіюється в output за правилом:

$output[r + 4c] = state[r, c]$, для $0 \leq r < 4$ і $0 \leq c < Nb$.

Допоміжні процедури:

AddRoundKey()	трансформація при шифруванні та зворотному шифруванні, при якій Round Key XOR'ється з State. Довжина RoundKey дорівнює розміру State
InvMixColumns()	трансформація при розшифруванні, яка є зворотною по відношенню до MixColumns()
InvShiftRows()	трансформація при розшифруванні, яка є зворотною щодо ShiftRows()
InvSubBytes()	трансформація при розшифруванні, яка є зворотною щодо SubBytes()
MixColumns()	трансформація при шифруванні, яка бере всі стовпці State і змішує їх дані (незалежно один від одного), щоб отримати нові стовпці
RotWord()	функція, що використовується в процедурі Key Expansion, яка бере 4-байтове слово і здійснює над ним циклічну перестановку
ShiftRows()	трансформації при шифруванні, що обробляють State, циклічно зміщуючи останні три рядки State на різні величини

SubBytes()	трансформації при шифруванні, що обробляють State, використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State
SubWord()	функція, що використовується у процедурі Key Expansion, яка бере на вході чотирибайтове слово i, застосовуючи S-box до кожного з чотирьох байтів, видає вихідне слово

SubBytes() процедура обробляє кожен байт стану, незалежно роблячи нелінійну заміну байтів, використовуючи таблицю замін (S-box). Така операція забезпечує нелінійність алгоритму шифрування. Побудова S-box складається із двох кроків. По-перше, провадиться взяття зворотного числа в полі Галуа GF(2^8). Для всіх операцій у цьому полі використовується поліном, що не наводиться $z^8 + z^4 + z^3 + z + 1$. По-друге, до кожного байта b, у тому числі складається S-box, застосовується така операція:$b'_i = b_i \oplus b_{(i+4)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+6)\ mod\ 8} \oplus b_{(i+7)\ mod\ 8} \oplus c_i$

де $0 \leq i < 8$, та де в b_i існує i-тий біт b, а c_i – i-тий ьіт константи $c = 63_{16} = 99_{10} = 01100011_2$. Таким чином забезпечується захист від атак, заснованих на простих властивостях алгебри.

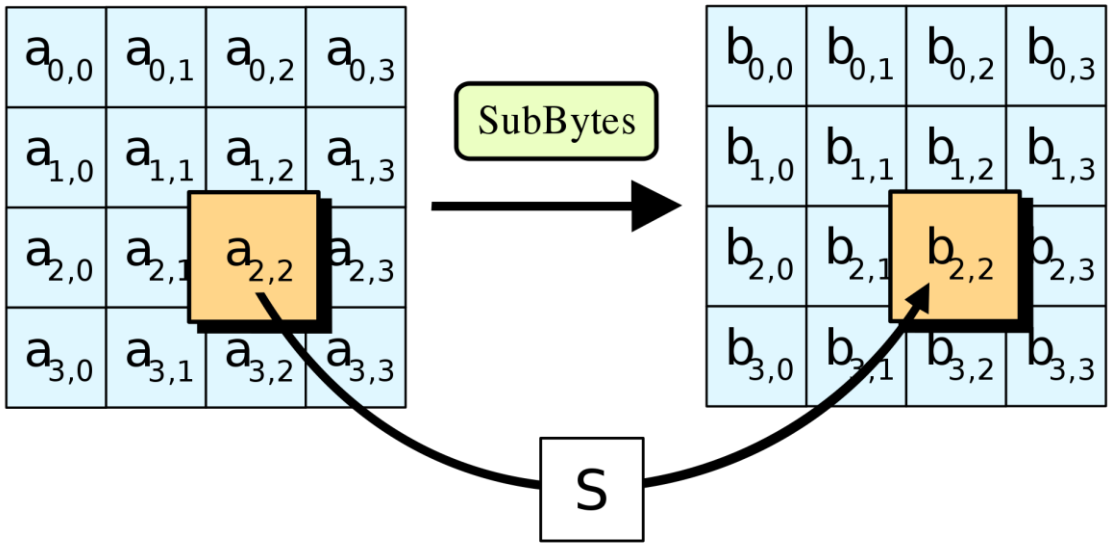

Рисунок 1.3.1 У процедурі SubBytes, кожен байт у state замінюється відповідним елементом у фіксованій 8-бітній таблиці пошуку.

ShiftRows працює зі рядками State. При цій трансформації рядки стану циклічно зсуваються на r байт по горизонталі, залежно від номера рядка. Для нульового рядка r = 0, для першого рядка r = 1 Б і т. д. Таким чином кожна колонка вихідного стану після застосування процедури ShiftRows складається з байтів з кожної колонки початкового стану. Для алгоритму Rijndael патерн зміщення рядків для 128- та 192-бітних рядків однаковий. Однак для блоку розміром 256 біт відрізняється від попередніх тим, що 2, 3 і 4 рядки зміщуються на 1, 3 і 4 байти відповідно. Це зауваження не стосується AES, тому що він використовує алгоритм Rijndael тільки з 128-бітними блоками, незалежно від розміру ключа .

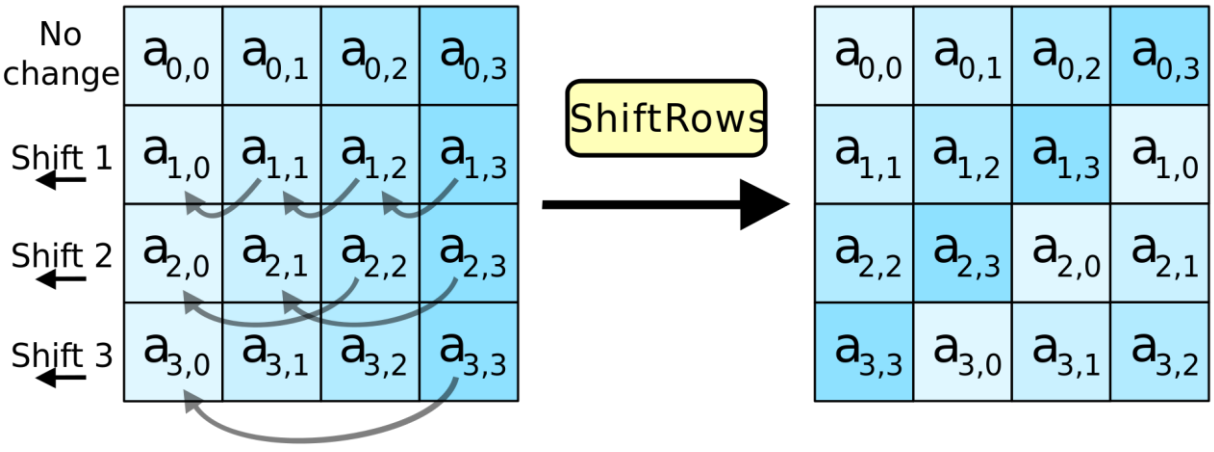

Рисунок 1.3.2 У процедурі ShiftRows байти у кожного рядка state циклічно зсуваються вліво. Розмір усунення байтів кожного рядка залежить від його номера

У процедурі MixColumns чотири байти кожної колонки State змішуються, використовуючи при цьому оборотну лінійну трансформацію. MixColumns обробляє стани по колонках, трактуючи кожну їх як поліном третього ступеня. Разом із ShiftRows MixColumns вносить дифузію до шифру.

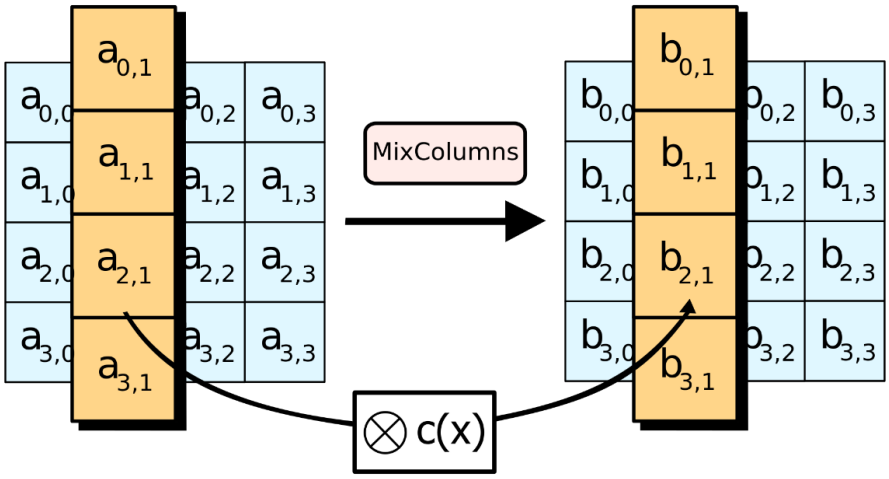

Рисунок 1.3.3 У процедурі MixColumns кожна колонка стану перемножується із фіксованим многочленом c(x).

У процедурі AddRoundKey RoundKey кожного раунду об'єднується зі State. Для кожного раунду Roundkey виходить із CipherKey за допомогою процедури KeyExpansion; кожен RoundKey такий самий розмір, як і State. Процедура здійснює побітовий XOR кожного байта State з кожним байтом RoundKey.

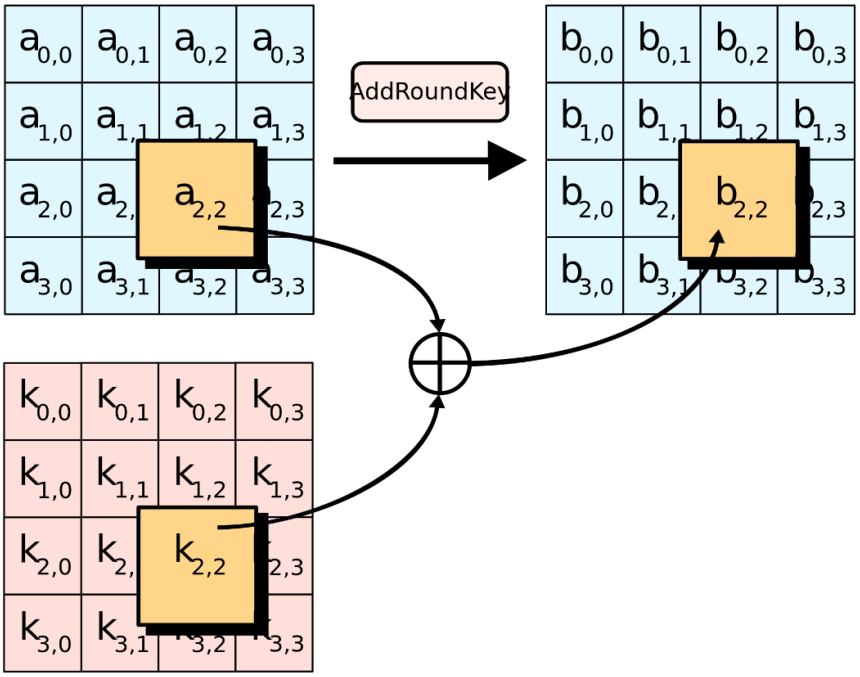

Рисунок 1.3.4 У процедурі AddRoundKey кожен байт стану поєднується з RoundKey, використовуючи операцію XOR

Алгоритм AES , генерації раундових ключів, використовуючи процедуру KeyExpansion() і подаючи Cipher Key, K, отримує ключі для всіх раундів. Усього виходить Nb*(Nr + 1) слів: спочатку алгоритму потрібен набір з Nb слів, і кожному з Nr раундів потрібно Nb ключових набору даних. Отриманий масив ключів для раундів позначається як $w[i], 0 \leq i < Nb * (Nr + 1)$. Функція SubWord() бере чотирибайтове вхідне слово та застосовує S-box до кожного з чотирьох байтів. Те, що вийшло, подається на вихід. На вхід RotWord() подається слово [a_0,a_1,a_2,a_3], яке вона циклічно переставляє та повертає [a_1,a_2,a_3,a_0]. Масив слів, постійний для даного раунду, Rcon[i], містить значення [x^{i-1},00,00,00], де x= {02}, а x^{i-1} це степінь x у GF(2^8) (i починається з 1). Перші Nk слів розширеного ключа заповнені Cipher Key. У кожне наступне слово, w[i], кладеться значення, отримане під час операції XOR w[i-1] та w[i-Nk]. Для слів, позиція яких кратна Nk, перед XOR'ом до w[i-1] застосовується трансформація, за якою слідує XOR з константою раунду Rcon[i]. Вказана вище трансформація складається з циклічного зсуву байтів у слові

(RotWord()), за якою слідує процедура SubWord() — те саме, що й SubBytes(), тільки вхідні та вихідні дані будуть розміром у слово.

Про крипостійкість цього алгоритму достатньо сказати, що у червні 2003 року Агентство національної безпеки США ухвалило, що шифр AES є достатньо надійним, щоб використовувати його для захисту відомостей, що становлять державну таємницю. До рівня SECRET було дозволено використовувати ключі довжиною 128 біт, для рівня TOP SECRET були потрібні ключі довжиною 192 і 256 біт.

1.3.3 UI Framework Avalonia

Avalonia – це кросплатформова реалізація WPF , яка перейняла його основні плюси та додала свої. Avalonia підтримує XAML, а отже проекти можливо розділяти на view та business logic. Фреймворк має класи та псевдокласи для написання стилізації , що значно спрощує написання стилів та їх розуміння. Якщо при створенні традиційних програм на основі WinForms за малювання елементів керування та графіки відповідали такі частини ОС Windows, як User32 та GDI+, то програми Avalonia засновані на DirectX. У цьому полягає ключова особливість рендерингу графіки Avalonia. Значна частина роботи з малюванням графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягати на графічний процесор на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.

Фреймворк було розроблено восени 2021 року, він постійно розвивається і на даний момент вже є підтримка таких операційних систем:

- Windows;
- Mac OS;
- Linux;
- iOS;
- Android;

Основні переваги Avalonia від WinForms та WPf:

- Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, що базується на xml і представляє альтернативу програмному створенню графіки та елементів управління, а також можливість комбінувати XAML та C#/VB.NET
- Незалежність від роздільної здатності екрана: оскільки у Avalonia всі елементи вимірюються в незалежних від пристрою одиницях, програми на Avalonia легко масштабуються під різні екрани з різною роздільною здатністю.
- Нові можливості, яких складно було досягти WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми та ін.
- Багаті можливості створення різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми та багато іншого
- Апаратне прискорення графіки - незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, а потім візуалізують за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіку більш плавною.
- Підтримка різних операційних систем – додатки написанні на Avlonia запускаються на Windows, Linux, Mac OS, iOS, Android.

Варто враховувати, що в порівнянні з програмами на Windows Forms обсяг програм на Avalonia і споживання ними пам'яті в процесі роботи в середньому дещо вище. Але це з лишком компенсується ширшими графічними можливостями та підвищеною продуктивністю при малюванні графіки.

1.3.4 .NET Framework

.NET Framework — це програмна платформа, випущена Microsoft у 2002 році. Платформа заснована на Common Language Runtime (CLR) для різних мов програмування. Функціональність CLR доступна на будь-якій мові програмування, яка використовує це середовище. Наразі .NET Framework розробляється як .NET.

.NET Framework вважався відповіддю Microsoft на популярну на той час платформу Java Sun Microsystems (тепер належить Oracle).

Хоча .NET Framework є власною технологією Microsoft і офіційно розроблена для роботи в операційних системах Windows, існують також незалежні проекти (переважно Mono і Portable.NET), які дозволяють запускати .NET Framework на деяких інших програмах операційних систем.

Програма для .NET Framework, написана будь-якою мовою програмування, що підтримується, спочатку перекладається компілятором в єдиний для .NET проміжний байт-код Common Intermediate Language (CIL) (раніше називався Microsoft Intermediate Language, MSIL). У термінах .NET виходить збирання, англ. assembly. Потім код або виконується віртуальною машиною Common Language Runtime (CLR), або транслюється утилітою NGen.exe виконуваний код для конкретного цільового процесора. Використання віртуальної машини переважно, тому що позбавляє розробників необхідності дбати про особливості апаратної частини.

У разі використання віртуальної машини CLR вбудований в неї JIT-компілятор "на льоту" (just in time) перетворює проміжний байт-код на машинні коди потрібного процесора. Сучасна технологія динамічної компіляції дозволяє досягти високого рівня швидкодії. Віртуальна машина CLR також сама дбає про базову безпеку, управління пам'яттю і систему виключень, позбавляючи розробника частини роботи.

Фреймворк .NET є потужною платформою для створення додатків. Можна виділити такі основні риси:

- Підтримка кількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C# це також VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код будь-якою з цих мов компілюється у складання загальною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET. Тому за певних умов ми можемо зробити окремі модулі однієї програми окремими мовами.
- Кросплатформність. .NET є платформою, що переноситься (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент – .NET 6 підтримується на більшості сучасних ОС Windows, MacOS, Linux. Використовуючи різні технології на платформі .NET, можна розробляти програми на мові C# для різних платформ - Windows, MacOS, Linux, Android, iOS, Tizen.
- Потужна бібліотека класів. .NET представляє єдину всім підтримуваних мов бібліотеку класів. І яку б програму ми не збиралися писати на C# - текстовий редактор, чат або складний веб-сайт - так чи інакше ми використовуємо бібліотеку класів .NET.
- Різноманітність технологій. Загальномовне середовище виконання CLR та базова бібліотека класів є основою цілого стеку технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних у цьому стеку технологій призначено технологію ADO.NET та Entity Framework Core. Для побудови графічних програм з багатим насиченим інтерфейсом - технологія WPF і WinUI, для створення більш простих графічних програм - Windows Forms. Для розробки кросплатформових

мобільних та десктопних програм - Xamarin/MAUI. Для створення веб-сайтів та веб-додатків - ASP.NET.

– Продуктивність. Відповідно до ряду тестів веб-програми на .NET 6 у ряді категорій сильно випереджають веб-програми, побудовані за допомогою інших технологій. Програми на .NET 6 у принципі відрізняються високою продуктивністю.

1.3.5 Середовище розробки

Microsoft Visual Studio — лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення та інші інструменти.

Visual Studio включає редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторингу коду. Вбудований налагоджувач може працювати як налагоджувач рівня вихідного коду, так і відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion та Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування та візуального проектування) коду предметно-орієнтованими мовами програмування) або інструментами для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

1.3.6 Архітектура MVVM

Спілкування користувача з системою відбувається за патерном – MVVM. MVVM(Model-View-ViewModel) дозволяє відокремити логіку

програми від візуальної частини (уявлення). Цей патерн є архітектурним, тобто він задає загальну архітектуру програми.

На рисунку 4.3.1 зображена схематична модель патерну MVVM.

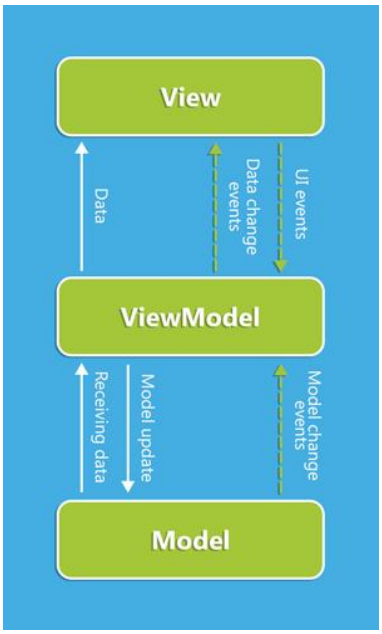

Рисунок 1.3.5 – Схематичне зображення MVVM

Модель описує дані, що використовуються в додатку. Моделі можуть містити логіку, безпосередньо пов'язану з цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити жодної логіки, пов'язаної з відображенням даних та взаємодією з візуальними елементами керування. Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомити систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до уявлення, хоча знову ж таки пряма взаємодія між моделлю і уявленням відсутня.

View або представлення визначає візуальний інтерфейс, через який користувач взаємодіє з програмою. Стосовно Avalonia уявлення - це код xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) у Avalonia може містити як інтерфейс у xaml, так і прив'язаний до нього код C#, проте в ідеалі код C# не повинен містити якоїсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent та виконує початкову ініціалізацію вікна . Все ж таки основна логіка програми виноситься в компонент ViewModel. Але коли в файлі пов'язаного коду вже може бути певна логіка, яку складно продати в рамках патерну MVVM в ViewModel. Подання не обробляє події за рідкісним винятком, а виконує дії переважно за допомогою команд.

ViewModel або модель уявлення пов'язує модель та уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна даних, що відображаються в поданні, хоча безпосередньо модель і уявлення не пов'язані. ViewModel також містить логіку з отримання даних з моделі, які потім передаються у виставу. І також VewModel визначає логіку щодо оновлення даних у моделі. Оскільки елементи уявлення, тобто візуальні компоненти типу кнопок, не використовують події, уявлення взаємодіє з ViewModel за допомогою команд. Наприклад, користувач хоче зберегти дані, що були введені в текстове поле. Він натискає кнопку і тим самим відправляє команду в ViewModel. А ViewModel вже отримує передані дані та відповідно до них оновлює модель.

Переваги MVVM – додатків:

- Програми, розроблені з використанням MVVM, мають дуже хорошу основу для проведення модульного тестування з метою перевірки роботи окремих класів і методів.
- Обсяг коду, необхідного для керування представленням, трохи знижується при використанні MVVM, а це означає, що знижується ризик припуститися помилок і зменшується код для написання модульних тестів.

– MVVM передбачає добре організовану та легку для розуміння конструкцію побудови графічного інтерфейсу за рахунок використання механізмів прив'язок, команд та шаблонів даних.
– Розробники та дизайнери можуть самостійно працювати над різними частинами програми.

Підсумком застосування патерну MVVM є функціональний поділ програми на три компоненти, які простіше розробляти та тестувати, а також надалі модифікувати та підтримувати.

1.4 Постановка технічного завдання

У ході аналізу предметної галузі було розглянуто проблему незахищеності користувацьких файлів, яка є досить актуальною в наш час, проведено дослідження додатків для шифрування даних, в ході якого було виявлено їх переваги та недоліки. Таким чином створення додатку для шифрування файлів з метою удосконалити їх захист є актуальним.

Було вирішено створювати кроспалтформовий додаток, щоб збільшити кількість людей які будуть ним користуватись, для таких операційних систем:

– Windows;
– Linux;
– macOS;

Додаток шифрування користувацьких файлів повинен забезпечувати наступні функції:

– Генерації ключа на основі паролю якій вводить користувач;
– Пошук файлів на електронному носії інформації для його подальшого використання;

– Шифрування файлів;
– Видалення незашифрованої копії з комп’ютера;
– Дешифрування файлу;
– Використання раніше згенерованих ключів;
– Зберігання згенерованого ключа.

Інструменти для реалізації програми повинні підтримувати ці операційні системи, вміти працювати с користувацькими файлами, підтримувати алгоритми шифрування. Для вибору інструментів був проведений аналіз засобів розробки , в ході якого були вибрані такі засоби:

– Мова програмування C#;
– Алгоритм шифрування AES;
– UI Framework Avalonia;
– Framework .Net.

2.РОЗРОБКА СТРУКТУРИ ДОДАТКУ ДЛЯ ШИФРУВАННЯ КОРИСТУВАЦЬКИХ ФАЙЛІВ

2.1 Завдання додатку для захисту користувацьких файлів

Зберігання важливих файлі на власному комп’ютері , який має доступ до інтернету , є дуже небезпечним , адже в любий момент електронний носій інформації може бути зараженим шкідливою програмою яка викраде ці дані. Вирішення цієї проблеми – це система , яка шифрує користувацькі файли та не дає зловмисниками отримати їх в початковому виді.

Основна перевага додатків – це захист інформації в файлах навіть якщо зловмисник зміг оволодіти ними. Використання таких додатків значно зменшує шанси викрадення особистих даних.

Проектована програма дозволить виконувати такі задачі, пов’язані з шифруванням файлів:

– Створення ключів шифрування;
– Створення зашифрованою копії файлу;
– Дешифрування файлу;
– Видалення незашифрованої копії файлу;
– Збереження публічного ключа шифрування;

Мета роботи - покращення захисту користувацьких файлів за рахунок використання криптографічного додатку на базі платформ .NET та Avalonia.

2.2 Моделювання об’єкту проектування

2.2.1 Діаграма прецендентів додатку

UML діаграма прецедентів узагальнює деталі користувачів системи та їх взаємодію з системою.

Для її побудови потрібно використовувати набір спеціалізованих символів та роз'ємів. Правильно побудована діаграма зображує графічне представлення :

- Сценарії взаємодій , в яких система чи додаток взаємодіють з людьми , організаціями чи зовнішніми системами;
- Цілі , які додаток чи система дає змогу досягти;
- Обсяг системи;

Суть діаграми прецедентів полягає в тому, що розроблена система представляється як набір сутностей або акторів, які взаємодіють із системою за допомогою так званих випадків використання. Варіанти використання використовуються для опису послуг, які система надає акторам. Іншими словами, кожен варіант використання визначає певний набір дій, які система виконує під час розмови з учасником. У той же час, про те, як буде реалізована взаємодія акторів із системою, говорити мало.

UML — це інструмент моделювання для побудови діаграм. Використовувана оболонка позначена овалом. Фігурки зображують акторів у процесі, а їхня участь у системі моделюється лінією між акторами та варіантами використання. Щоб намалювати межі системи, вам потрібно намалювати поле навколо варіанта використання. Загальні компоненти діаграми прецедентів:

- Актори – це користувачі системи. Актором може бути людина, організація або зовнішня система. Загалом актором може бути будь-який зовнішній об'єкт , який виробляє або споживає данні та взаємодіє з системою.

- Система – сценарій послідовних дій та взаємодій між актором та системою.
- Цілі – це кінцевий результат випадків використання. Діаграма повинна описувати діяльність та варіанти , які використовуються для кінцевої мети
- Кейси використання – представляють собою овали в горизонтальній формі , які показують види використання користувачем.
- Асоціації – це лінія між учасниками та випадками використання.
- Системні граничні поля – це так звана коробка яка встановлює область системи для використання справ. Усі випадки використання поза межами цієї коробки розглядаються за межами цієї системи.

Додаток для шифрування користувацьких файлів розроблений як проект з відкритим кодом , в нього немає регестрації чи додаткових платних функцій, тому програма містить лише одного актора – це сам користувач. Мета цього проекту надати користувачу необхідний мінімум для захисту файлів методом їх шифрування. На Рисунку 2.2.1 показані основні прецеденти додатку.

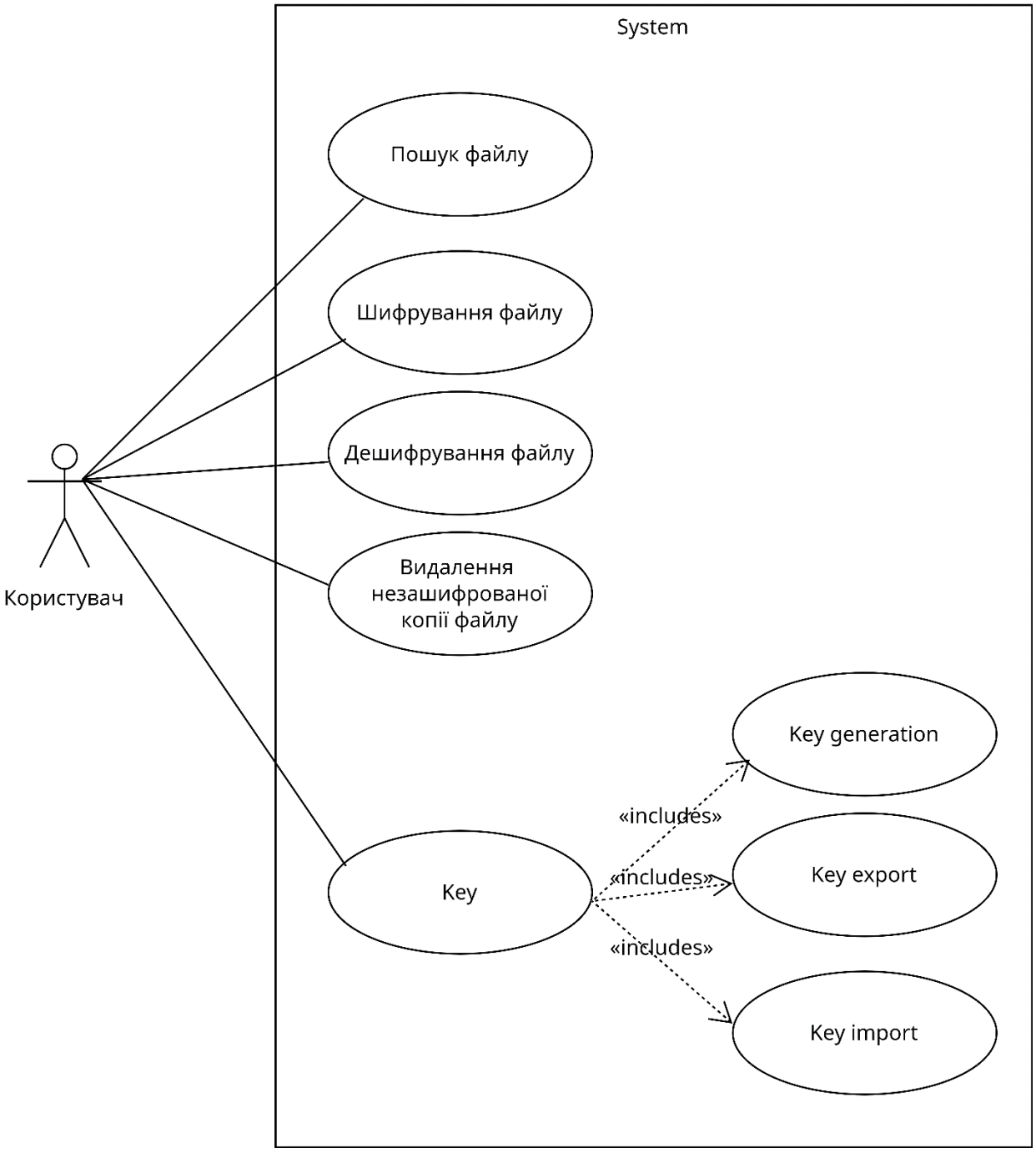

Рисунок 2.2.1 – UML Діаграма прецедентів додатку

2.2.2 Діаграма діяльності

UML діаграма діяльності – відображає динамічні аспекти поведінки системи. Вона використовується для відображення послідовності дій додатку. На Рисунку 2.2.2 зображений основний робочий процес додатку.

Дана діаграма показує робочий процес програми від початку до кінця, докладно описуючи способів послідовності подій , що містяться в дії.

Коли користувач запускає додаток , йому потрібно створити ключ для шифрування або дешифрування. Він може створити новий або використати раніше згенерований. Ключ передається до алгоритмів шифрування або дешифрування , в залежності від вибору користувача, після чого вибирається файл для подальшого його використання. Останніми етапами роботи програми є шифрування або дешифрування файлу , після чого основна роботи програми є закінченою.

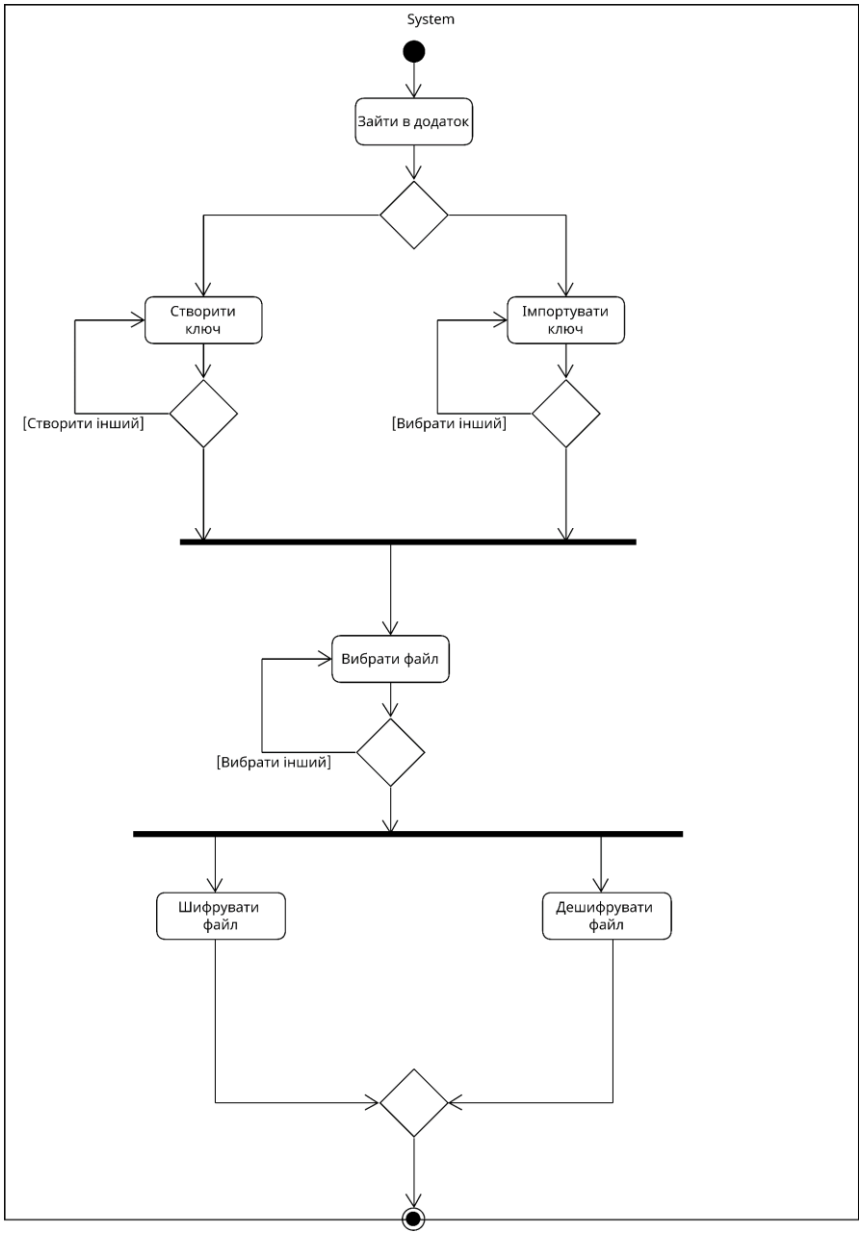

Рисунок 2.2.2 – UML діагарма діяльності додатку

3.ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Розробка концепції

Аналізуючи аналоги та актуальність десктопного додатку, для ефективного розповсюдження було прийняте рішення розробляти його для різних операційних систем , а саме:

- Windows;
- Linux;
- MacOS;

Цільова аудиторія – користувачі цих ОС які мають важливі файли або які просто хочуть захисти себе від зловмисників.

Короткий опис застосування – десктопний додаток націлений на захист користувацьких файлів. За його допомогою користувач може керувати доступом до файлів , шляхом їх шифрування. Доступ до файлів будуть мати тільки користувачі які мають ключ яким було зашифровані дані. Також користувач зможе створювати ключі шифрування на основі веденого паролю та зберігати їх , для подальшого використання.

Головні вимоги додатку:

- Створення ключа шифрування;
- Збереження ключа шифрування;
- Імпорт ключа шифрування;
- Шифрування даних;
- Видалення незашифрованої копії файлу;
- Дешифрування даних;
- Пошук файлів для шифрування;

– Візуальна частина додатку;

3.2 Структура розробленого додатку

Система шифрування користувацьких файлів буде складатися з п’яти модулів, кожен з яких буде розбиватись на функціональні підблоки. Головними модулями системи будуть алгоритм шифрування та дешифрування даних. На рисунку 4.1 зображена схема цих модулів та їх зв’язок.

Для всіх необхідних модулів та задач системи було проаналізовано варіанти рішень , серед яких було обрані ті що задовольняють наступні умови:

– Швидкодія алгоритму;
– Безпечність використання;
– Безперешкодна інтеграція підсистем між собою;

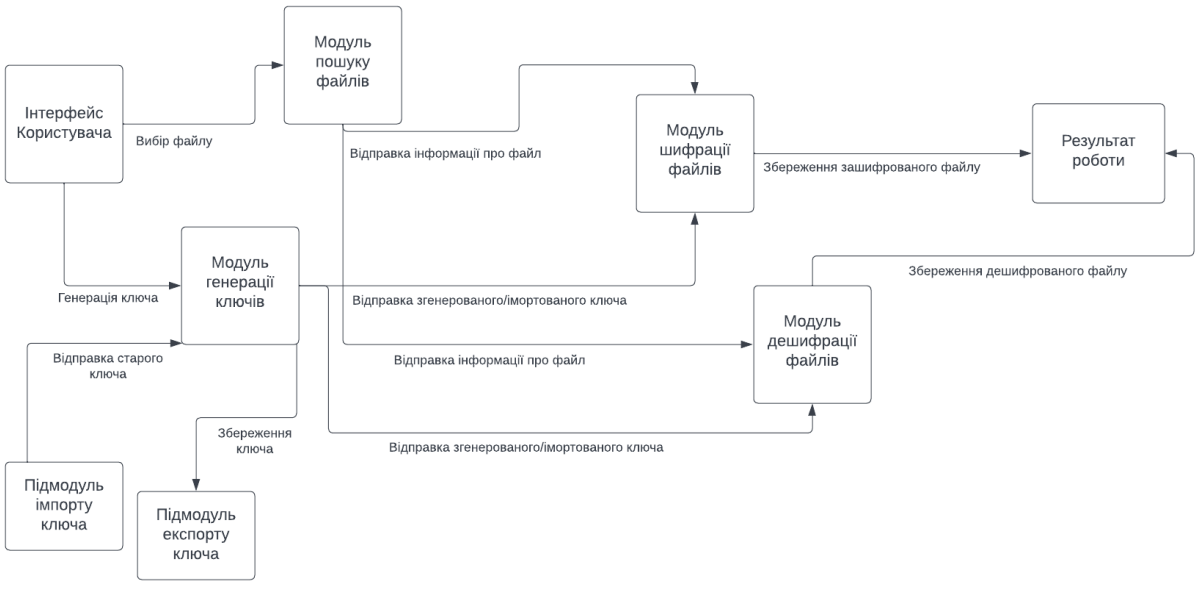

Рисунок 3.2.1 – Схематичне зображення роботи системи

Наприклад для того щоб не роботи інформаційний безлад на головному екрані додатку , був розроблений сторінковий інтерфейс додатку , який

дозволяє користувачу спокійно переходити між сторінками і бачити тільки саму необхідну інформацію.

Для того щоб користувач міг працювати з вже зашифрованими даними був розроблений модуль дешифрації файлів , та два підмодулі до генерації ключів, а саме підмодулі експорту та імпорту ключів, щоб користувач зміг використовувати та зберігати створенні ключі.

3.3 Набір інструментів та програмних засобів які використовуються для розробки

Додаток для захисту користувацьких файлів розробляється для персональних комп'ютерів. Найважливішим для даного виду програм є вибір алгоритму шифрування. Від нього залежить швидкість шифрування файлів, надійність та можливість обробляти великий обсяг даних. Алгоритми шифрування діляться на три типи :

- Симетричний – процес в якому ключ на основі якого виконується шифрування та дешифрування даних. Алгоритми даного типу є швидкими , невимогливими до апаратної складової та можуть ефективно обробляти великий обсяг даних;
- Асиметричний - це вид шифрування в якому поєднанні два види ключів: відкритого і закритого. За допомогою відкритого ключу проводиться шифрування ключової інформації перед відправленням, а вже в цільових вузлах вона дешифрується за вдяки приватному ключу який йде в математичній зв'язці з публічним. Даний алгоримт забезпечую безпечну передачу даних між двома комп'ютерами , але погано справляється з більшим об'ємом даних;
- Гібридний – це симбіоз двох попередніх алгоритмів , який перейняв всі їх переваги. В результаті поєднання отримали систему яка завдяки

асинхронної частини проводить аутентифікацію вузлів та допомагає безпечно обмінятися ключами симетричного шифрування.

В даній роботі розробляється додаток під системи Windows та Linux. В першу чергу для реалізації шифрування був обраний алгоритм шифрування AES. Це симетричний алгоритм блочного шифрування, з розміром блоку 128-біт та ключу 128/198/256-біт. Алгоритм є прийнятим стандартом шифрування в США, а це говорить про те що алгоритм є добре проаналізованим та надійним.

В якості мови розробки було обрано C#. Це об’єктно-орієнтована мова програмування, яка дозволяє розробляти різні типи додатків на різних платформах .NET. C# — сучасна мова, яка дозволяє відокремити обробку інтерфейсу користувача від бізнес-логіки вашої програми. Автоматичне керування пам'яттю шляхом збирання сміття дозволяє гнучко використовувати складні структури даних. .NET з підтримкою C# має готову бібліотеку для реалізації криптографічних алгоритмів, що значно економить час при розробці додатків.

Вибраний кросплатформний графічний інтерфейс XAML Framework Avalonia для створення графічних інтерфейсів. Це кросплатформна реалізація WPF, яка успадковує його основні переваги та додає власні. Avalonia підтримує XAML, тому проекти можна розділити на представлення та бізнес-логіку. У фреймворку є класи та псевдокласи для стилів письма, що значно спрощує стилі написання та їх розуміння.

3.4 Робота користувача з додатком

При запуску програми користувач попадає на головну сторінку додатку де вказані головні сторінки та за що вони відповідають, Рисунок 3.4.1.

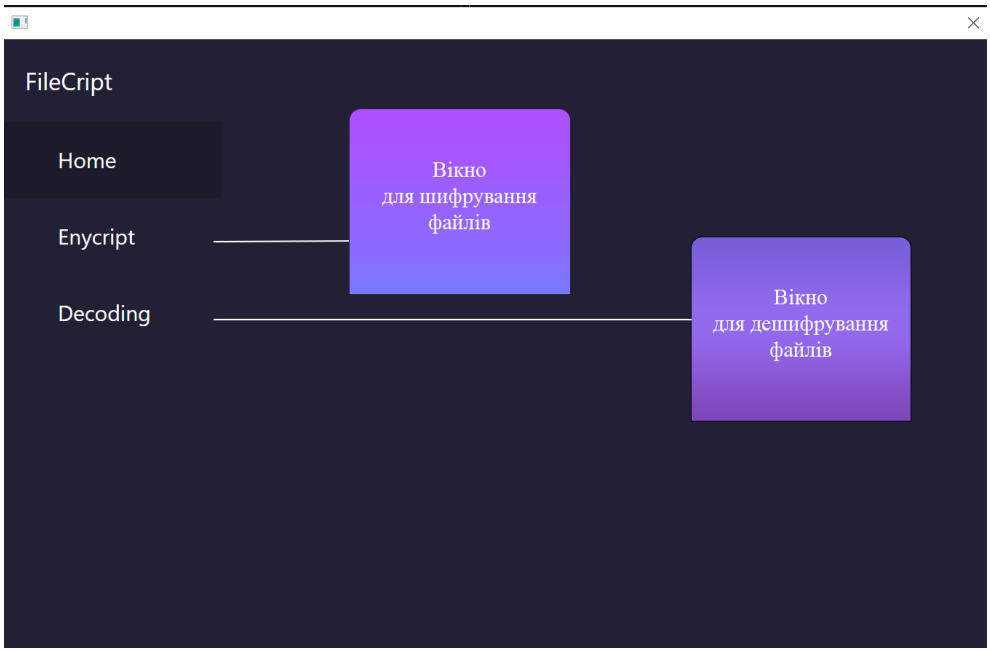

Рисунок 3.4.1 – Вікно при запуску додатку

Далі юзер переходить до сторінки Enycript. Де в полі створення ключа він вводить пароль на основі якого створюється ключ шифрування. При натисненні кнопки Serch , відкривається вікно вибору файлу. Після вибору файлу та створення ключа , користувач повинен натиснути на кнопку Enycript і вибраний файл буде зашифровано, Рисунок 3.4.2 та 3.4.3.

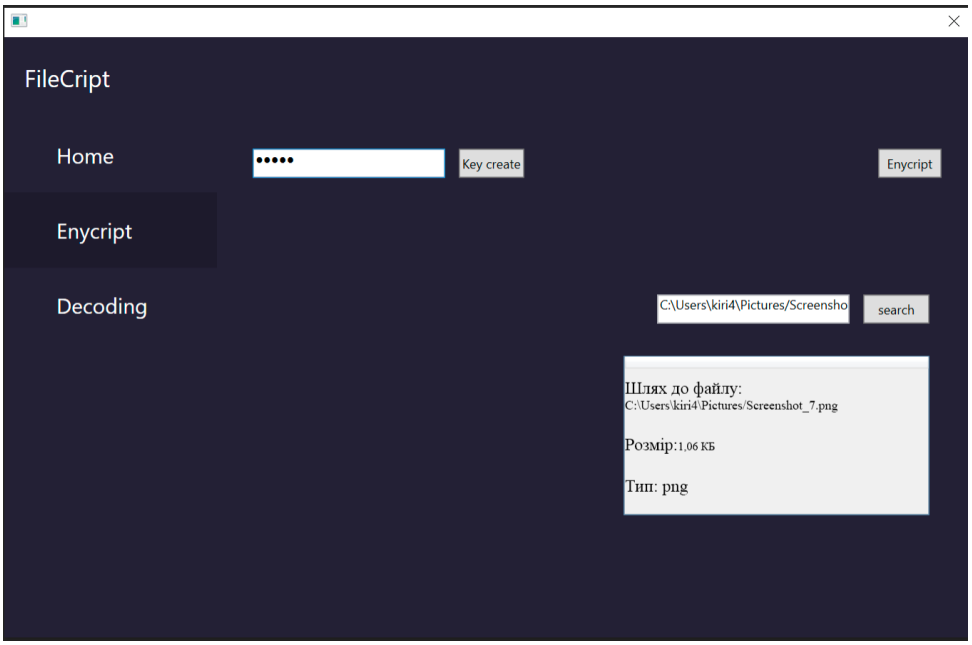

Рисунок 3.4.2 – Вікно шифрування файлу

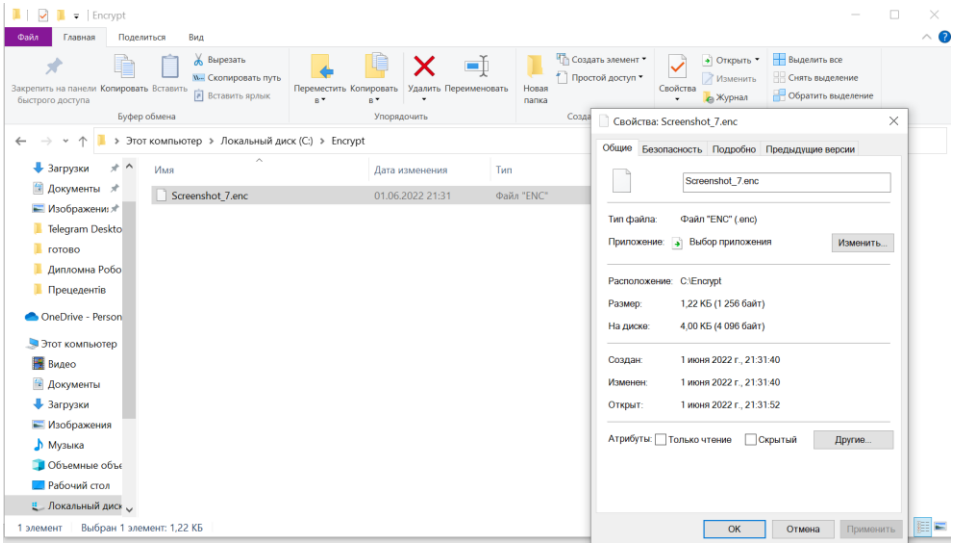

Рисунок 3.4.3 – Кінцевий результат шифрування файлу

Для доступу до цього файлу потрібно перейти на сторінку дешифрування та виконати ті самі дії , що й при шифруванні файлу, але пароль потрібно вводити такий же який був введений для кодування, та вибрати файл який зашифрували раніше, Рисунок 3.4.4 та 3.4.5.

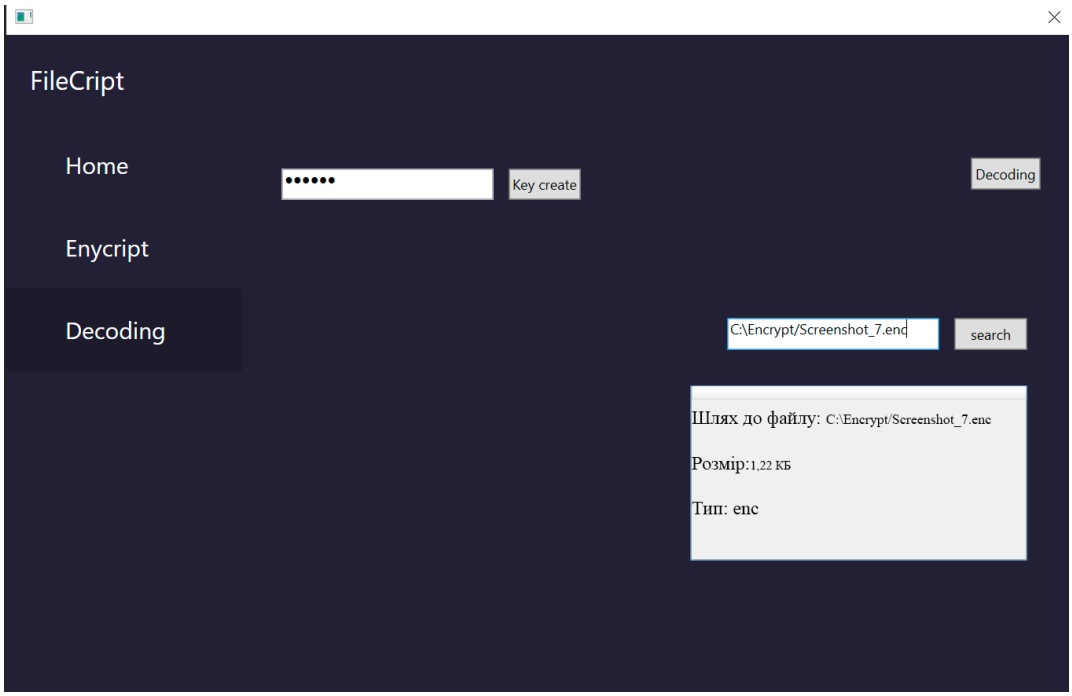

Рисунок 3.4.4 – Вікно дешифрування файлу

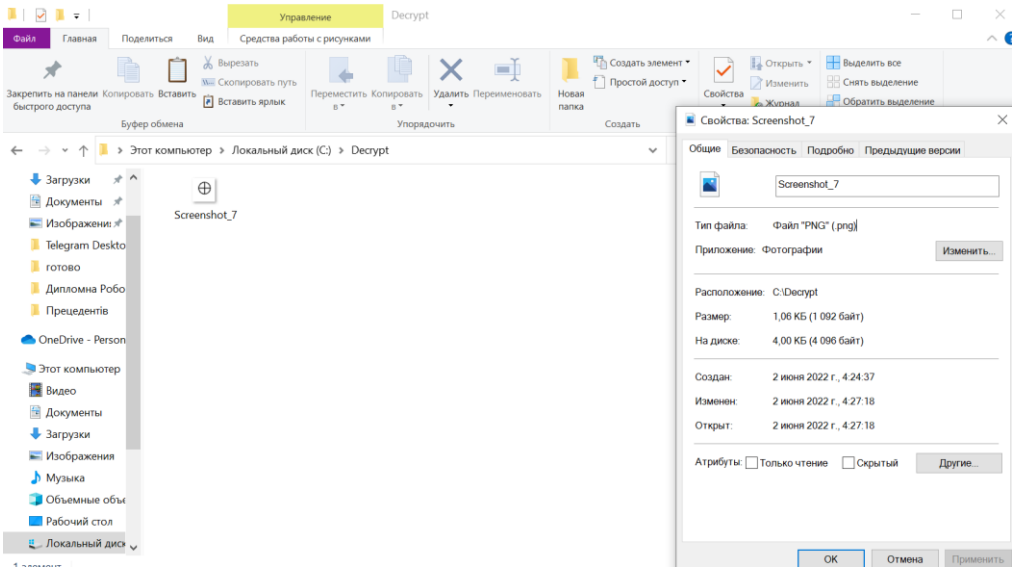

Рисунок 3.4.5 – Результат роботи дешифрування

3.5 Висновок до розділу

У даному розділі було розглянуто програмну реалізацію системи шифрування користувацьких файлів. Структура системи, приклад розв’язання поставлених задач, архітектура додатку, послідовність роботи користувача с додатком.

ВИСНОВКИ

У Під час роботи над дипломним проектом була розроблена програма для шифрування файлів користувача. Проаналізовано існуюче програмне забезпечення для вирішення поставленого завдання. Аналіз показує, що існуючі програми не повністю вирішують проблему незахищеності файлів користувачів. Деякі мають перевантажений і застарілий інтерфейс, який лякає непідготовлених користувачів.

У процесі розробки розробляються всі необхідні системи та підсистеми програми. Вибираються найкращі засоби розробки та методи проектування. Програма має монолітну архітектуру MVVM.

Розроблена програма дозволяє швидко та безпечно шифрувати необмежену кількість файлів усіх типів. Це повністю вирішує проблему користувачів, які бажають захистити свої дані від третіх осіб. Програмне забезпечення доступне для операційних систем: Windows, Linux, MacOS.

Потенційними користувачами програмного забезпечення є люди які не мають знань в алгоритмах шифрування даних, але хочуть захистити свою конфіденційну інформацію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В Україні кількість інтернет-користувачів у 2019 році збільшилась на 8% [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukrinform.ua/rubric-technology/2797152-v-ukraini-kilkist-internetkoristuvaciv-zrosla-do-23-miljoniv.html
2. Країни-жертви та країни-агресори у хакерських війнах [Електронний ресурс] – Режим доступу до ресурсу: https://www.slovoidilo.ua/2021/10/22/infografika/svit/krayiny-zhertvy-ta-krayiny-ahresory-xakerskyx-vijnax
3. The Hidden Costs of Cybercrime [Електронний ресурс] – Режим доступу до ресурсу: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-hidden-costs-of-cybercrime.pdf
4. Алгоритм блочного симетричного шифрування Advanced Encryption Standard (AES)[Електронний ресурс] – 2009 – Режим доступу до ресурсу: http://crypto.pp.ua/wp-content/uploads/2010/03/aes.pdf.
5. Патерн MVVM [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/wpf/22.1.php
6. CipherShed [Електронний ресурс] – Режим доступу до ресурсу: https://www.ciphershed.org/

   TrueCrypt [Електронний ресурс] – Режим доступу до ресурсу: https://www.gbls.org/sites/default/files/2018-02/TruCrypt%20Tutorial.pdf.
7. BitLocker [Електронний ресурс] / Microsoft – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-overview.
8. UML для бізнес-моделювання: навіщо потрібні діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ru/articles/uml-diagrams.html

9. Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C (cloth) [Електроний ресурс] / Bruce Schneier – 1996 – Режим доступу до ресурсу: https://doc.lagout.org/network/3_Cryptography/Applied%20Cryptography%2C%202nd%20Edition.pdf

10. AES Encryption: Study & Evaluation [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/346446212_AES_Encryption_Study_Evaluation

11. Theory and Implementation of Data Encryption Standard: A Review [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/45949430_Theory_and_Implementation_of_Data_Encryption_Standard_A_Review

12. Шпаргалка тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalearning.com.ua/theory/about_qa/shpargalka-z-testuvannya/

13. Основи тестування програмного забезпечення [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/ru/baza-znaniy/

14. Research on the Application of Data Encryption Technology Based on Network Security Maintenance in Computer Network Security [Електронний ресурс]. – Режим доступу: https://iopscience.iop.org/article/10.1088/1742-6596/1744/2/022060/pdf