МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

ПРОГРАМНЕ СЕРЕДОВИЩЕ ДЛЯ ШИФРУВАННЯ / ДЕШИФРУВАННЯ ІНФОРМАЦІЇ

Виконав: студент групи 1К-21
Спеціальності 123 Комп’ютерна інженерія
Тимофій БОРИСОВ
Керівник:
Наталя ФАЛЬЧЕНКО

Черкаси 2025

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню та практичній реалізації програмного середовища для шифрування/дешифрування інформації. В умовах зростання кіберзагроз та потреби в конфіденційності даних, питання захисту інформації набуває особливої актуальності, а криптографічні методи стають ключовими інструментами забезпечення інформаційної безпеки. Загальний обсяг роботи 51 сторінка, містить 16 рисунків, 6 таблиць, 6 формул та посилається на 26 джерел.

У роботі проведено аналіз сучасних вимог до програм захисту даних, розглянуто основні принципи шифрування, включаючи симетричні та асиметричні алгоритми, а також сучасні розробки в криптографії, такі як постквантова та гомоморфна криптографія. Детально охарактеризовано класичний поліалфавітний шифр Бофорта, обраний для практичної реалізації, та проведено його порівняння з іншими класичними шифрами (Хілла, Віженера). Здійснено огляд існуючих програмних рішень для шифрування файлів, повідомлень, трафіку та хмарних сховищ (EFS, VeraCrypt, BitLocker, Signal, ProtonMail, WireGuard, Cryptomator та інші), що дозволило визначити переваги та недоліки сучасних систем захисту.

Ключовою частиною роботи є розробка програмного середовища RPP.

Описано архітектуру та логіку функціонування розробленої програми, що забезпечує виконання операцій шифрування/дешифрування тексту за допомогою шифру Бофорта. Представлено діаграму кінцевого автомату середовища RPP, що візуалізує послідовність взаємодії користувача з програмою.

Практична цінність роботи полягає в розробці функціонального програмного засобу, що демонструє принципи симетричного шифрування. Проведено тестування розробленого середовища RPP, результати якого підтверджують коректність його роботи та ефективність реалізованого алгоритму.

Ключові слова: інформаційна безпека, шифрування, дешифрування, криптографія, шифр Бофорта, Python, програмне середовище, RPP, захист даних.

ABSTRACT

The qualification work is devoted to the research and practical implementation of a software environment for encrypting/decrypting information. In the context of growing cyber threats and the need for data confidentiality, the issue of information protection is becoming particularly relevant, and cryptographic methods are becoming key tools for ensuring information security. The total volume of the work is 51 pages, contains 16 figures, 6 tables, 6 formulas and refers to 26 sources.

The work analyzes modern requirements for data protection programs, considers the basic principles of encryption, including symmetric and asymmetric algorithms, as well as modern developments in cryptography, such as post-quantum and homomorphic cryptography. The classical polyalphabetic Beaufort cipher, selected for practical implementation, is characterized in detail, and its comparison with other classical ciphers (Hill, Vigenère) is carried out. A review of existing software solutions for encrypting files, messages, traffic and cloud storage (EFS, VeraCrypt, BitLocker, Signal, ProtonMail, WireGuard, Cryptomator and others) was carried out, which allowed us to identify the advantages and disadvantages of modern protection systems.

The key part of the work is the development of the RPP software environment.

The architecture and logic of the functioning of the developed program is described, which ensures the execution of text encryption/decryption operations using the Beaufort cipher. A state machine diagram of the RPP environment is presented, which visualizes the sequence of user interaction with the program.

The practical value of the work lies in the development of a functional software tool that demonstrates the principles of symmetric encryption. The developed RPP

environment was tested, the results of which confirm the correctness of its operation and the effectiveness of the implemented algorithm.

Keywords: information security, encryption, decryption, cryptography, Beaufort cipher, Python, software environment, RPP, data protection.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій та широким використанням цифрових даних у різних сферах діяльності. В умовах глобалізації та активного поширення мережевих технологій питання захисту інформації набуває особливої актуальності[4].

Кіберзлочинність, несанкціонований доступ до даних, викрадення особистої та комерційної інформації – усі ці загрози вимагають ефективних рішень для забезпечення безпеки інформації. Без належного рівня захисту дані можуть бути змінені, викрадені або навіть знищені, що може призвести до значних фінансових та репутаційних втрат як для організацій, так і для окремих користувачів.

Одним із ключових методів захисту інформації є її шифрування[4]. Це процес перетворення вихідних даних у зашифрований формат, який може бути відновлений лише за допомогою спеціального ключа. Завдяки шифруванню можна гарантувати конфіденційність, цілісність та автентифікацію даних, що є надзвичайно важливим для державних, комерційних та приватних інформаційних ресурсів. Використання сучасних криптографічних методів дозволяє не тільки запобігти витоку даних, а й забезпечити їх захист у разі перехоплення зловмисниками.

Метою даної роботи є розробка програмного середовища для шифрування та дешифрування інформації, яке забезпечуватиме високий рівень безпеки та зручність у використанні[4]. В рамках роботи проаналізовано основні криптографічні алгоритми, визначено їх переваги та недоліки, а також реалізовано програмне рішення, яке дозволить ефективно здійснювати операції з шифрування та дешифрування даних. Особливу увагу приділено питанням оптимізації алгоритмів шифрування з метою підвищення швидкодії програмного забезпечення без зниження рівня безпеки.

Актуальність теми зумовлена необхідністю створення надійних програмних засобів для захисту інформації у різних сферах діяльності, зокрема у бізнесі,

державному управлінні, банківській сфері та приватному секторі[4]. Запропоноване програмне середовище сприятиме підвищенню рівня безпеки інформаційних систем та зменшенню ризиків несанкціонованого доступу до конфіденційних даних. Крім того, зростання вимог до безпеки цифрових даних у зв'язку з посиленням законодавчого регулювання захисту інформації робить розробку подібних систем ще більш актуальною.

Об'єктом дослідження є система забезпечення інформаційної безпеки шляхом використання криптографічних методів.

Предметом дослідження є технології шифрування і дешифрування даних, а також способи його їх реалізації у програмному середовищі.

Структура кваліфікаційної роботи включає аналіз існуючих методів шифрування, обґрунтування вибору алгоритму, опис процесу розробки програмного середовища та його тестування, а також підсумкові висновки щодо ефективності розробленого рішення. Дослідження охоплює вивчення популярних криптографічних алгоритмів, таких як AES, RSA, DES та інші, їх порівняння та визначення оптимального варіанту для впровадження у програмне середовище[2]. Крім того, буде розглянуто аспекти реалізації безпечного зберігання ключів та управління доступом до зашифрованих даних, що дозволить підвищити рівень захисту системи в цілому[2], [12], [13].

РОЗДІЛ 1 СУЧАСНІ ВИМОГИ ДО ПРОГРАМ ЗАХИСТУ ДАНИХ

1.1 Основні принципи шифрування даних

Ключовими принципами інформаційної безпеки, що забезпечуються за допомогою сучасних криптографічних методів, є такі[4], [1].

1. Конфіденційність (Confidentiality) полягає у тому, що доступ до даних мають лише авторизовані особи. Якщо повідомлення буде перехоплено сторонньою особою, вона не зможе його прочитати без спеціального ключа. Це досягається шляхом шифрування, коли інформація перетворюється у вигляд, що не має сенсу для сторонніх осіб (наприклад, текст «Пароль123» може перетворитися на «r3H7z!0p@»). Для цього застосовуються алгоритми шифрування, такі як AES, RSA, ChaCha20 тощо[12], [11].

2. Цілісність (Integrity) гарантує, що дані не були змінені або пошкоджені під час зберігання або передачі. Вона забезпечується за допомогою хеш-функцій, наприклад, SHA-256[24]. Дані пропускаються через хеш-алгоритм, в результаті чого створюється унікальний "відбиток" (hash). Якщо змінити бодай одну літеру в повідомленні, хеш значно зміниться, що дозволяє швидко виявити будь-яке втручання.

3. Автентифікація (Authentication) — це процес перевірки особи відправника або отримувача, щоб переконатися, що вони дійсно є тими, за кого себе видають. Автентифікація реалізується за допомогою логінів і паролів, а також цифрових підписів. Цифровий підпис — це зашифрований хеш повідомлення, який дозволяє будь-кому, хто має відкритий ключ, перевірити справжність підпису та цілісність повідомлення[11].

4. Контроль доступу (Access Control) визначає, хто і до чого має доступ. Це ,включає розмежування прав доступу за ролями — наприклад, адміністратор, користувач чи гість. Часто контроль доступу поєднується з шифруванням: навіть

якщо стороння особа отримає доступ до файлу, вона не зможе його прочитати без правильного ключа дешифрування.

Існує два основних типи шифрування: симетричне та асиметричне. При симетричному шифруванні використовується один і той самий ключ для шифрування і дешифрування даних[2]. Воно є швидким та простим у реалізації, але має недолік — складність безпечної передачі ключа. Прикладами таких алгоритмів є AES і DES[2], [12], [13]. Асиметричне шифрування базується на парі ключів: відкритому (public) і закритому (private). Дані шифруються відкритим ключем отримувача, і тільки він може розшифрувати їх своїм приватним ключем. Такий підхід забезпечує безпечну передачу даних без необхідності попереднього обміну ключами, хоча і є повільнішим за симетричне. Прикладами асиметричних алгоритмів є RSA та ECC[11].

Усі ці механізми разом забезпечують надійний захист даних від несанкціонованого доступу, змін і підробки, дозволяючи зберігати конфіденційність, точність і достовірність інформації в цифровому середовищі.

1.2 Характеристика алгоритмів дешифрування

Алгоритм декодування – це набір процедур, що використовуються для перетворення зашифрованих даних назад у їх оригінальний, зрозумілий вигляд[24]. Він фактично зворотній до процесу шифрування, дозволяючи авторизованим користувачам безпечно отримати доступ до інформації. Цей алгоритм призначений для забезпечення конфіденційності та цілісності даних, перетворюючи їх з нерозбірливого формату в оригінальний формат.

1.2.1 Принципи роботи алгоритмів декодування

Алгоритми декодування працюють у тісному зв’язку з алгоритмами шифрування, утворюючи єдину систему захисту інформації. Основна їхня мета — перетворити зашифровані дані (шифротекст) назад у початковий, зрозумілий формат, але лише за умови, що у користувача є відповідний ключ.

У загальному випадку процес декодування проходить кілька основних етапів. Спершу під час шифрування вихідні дані перетворюються у шифротекст — набір символів, що не має сенсу без спеціального алгоритму. Це здійснюється за допомогою певного методу шифрування та ключа, який відіграє вирішальну роль у забезпеченні захисту. Для зворотного перетворення авторизований користувач повинен мати доступ до алгоритму декодування та точно того ж ключа, що використовувався при шифруванні. Сам алгоритм виконує обчислення, які є зворотніми до тих, що застосовувались під час шифрування. Він використовує математичні функції, операції перестановок, підстановок, логічні операції та інші техніки для перетворення шифротексту назад у зрозумілу форму. У результаті користувач отримує початкові дані, придатні для читання, аналізу або подальшого використання. При цьому найважливішим аспектом є те, що лише особи, які мають правильний ключ, можуть провести цей процес — без нього інформація залишається зашифрованою та недоступною. Такий механізм забезпечує цілісність, конфіденційність і контроль доступу до важливих даних, що є критично важливим у цифровому середовищі.

1.2.2 Поради щодо запобігання несанкціонованому доступу до конфіденційної інформації

Щоб забезпечити ефективність алгоритму декодування та захистити конфіденційну інформацію, важливо дотримуватись таких принципів:

- Використовувати надійні, індустріально-стандартні алгоритми шифрування:
- Використовуйте алгоритми шифрування, які широко визнані та надійні у плані безпеки. Прикладами широко використовуваних алгоритмів шифрування є Advanced Encryption Standard (AES), RSA та Triple Data Encryption Standard (3DES) [12], [11], [13].
- Застосовувати багатофакторну аутентифікацію: Впровадження багатофакторної аутентифікації додає додатковий рівень безпеки, вимагаючи додаткових кроків перевірки поза шифруванням. Це допомагає запобігти несанкціонованому доступу до даних, навіть якщо шифрування якось було скомпрометоване.
- Регулярно оновлюйте протоколи і ключі шифрування: Вкрай важливо бути в курсі останніх протоколів шифрування та регулярно оновлювати ключі шифрування. Такий проактивний підхід допомагає захистити від нових загроз і вразливостей.

1.2.3 Приклади алгоритмів декодування

Прикладами алгоритмів декодування є такі[2], [24]:

1.Advanced Encryption Standard (AES) – один з найчастіше використовуваних алгоритмів шифрування та декодування. Він підтримує довжини ключів 128, 192 і 256 біт і широко визнаний за свою безпеку та ефективність. AES став стандартним алгоритмом шифрування для забезпечення безпеки даних у різних додатках, включаючи бездротові мережі, фінансові транзакції та урядові комунікації[12].

2.RSA – це ще один популярний алгоритм шифрування та декодування, що використовується у криптографії з відкритим ключем. Він отримав назву на честь своїх винахідників: Рона Рівеста, Аді Шаміра та Леонарда Адлемана. RSA використовує складність факторизації великих простих чисел для забезпечення

безпеки. Він широко використовується у протоколах безпечного зв'язку, цифрових підписах та шифруванні конфіденційних даних. [11]

3.Triple Data Encryption Standard (3DES) – симетричний алгоритм шифрування, оснований на початковому стандарті Data Encryption Standard (DES). Він застосовує алгоритм DES тричі, використовуючи два або три унікальні ключі. 3DES підвищує безпеку початкового алгоритму DES, збільшуючи розмір та складність ключа. Хоча AES значною мірою замінив DES та 3DES у нових впровадженнях, 3DES все ще використовується у деяких старих системах та додатках[13].

1.2.4 Останні розробки в сфері криптографії та інформаційної безпеки

Із розвитком технологій постійно з’являються нові алгоритми шифрування та декодування, які спрямовані на підвищення безпеки інформації та адаптацію до нових загроз. Однією з найактуальніших галузей є криптографія постквантового періоду[26]. У зв’язку з розвитком квантових обчислень виникає потреба у таких алгоритмах шифрування, які здатні витримати атаки з боку квантових комп’ютерів. Постквантова криптографія зосереджена на створенні методів захисту, що залишаються ефективними навіть після появи квантових обчислювальних потужностей, здатних зламувати сучасні шифри.

Ще однією революційною розробкою є гомоморфне шифрування — техніка, що дозволяє виконувати обчислення над зашифрованими даними без необхідності їх розшифровки[25]. Це відкриває нові можливості для безпечного аналізу даних у хмарних обчисленнях, зберігаючи при цьому конфіденційність інформації.

У свою чергу, технологія блокчейн також активно використовує шифрування — для забезпечення цілісності, незмінності та автентичності інформації в децентралізованих мережах застосовуються хеш-функції, цифрові підписи та інші криптографічні механізми[24].

Попри значні переваги, використання алгоритмів шифрування супроводжується низкою дискусій і викликів. Одним із ключових питань є проблема урядового спостереження: надійне шифрування може обмежувати можливості спецслужб щодо моніторингу та збору даних, що створює напругу між захистом особистої конфіденційності та забезпеченням національної безпеки. У цьому контексті виникають суперечки щодо доцільності впровадження так званих «бекдорів» — спеціальних механізмів, які дозволяють уповноваженим органам обходити шифрування у виняткових випадках. Противники цього підходу вказують на ризики зловживань та створення вразливостей, які можуть бути використані зловмисниками. Іншим важливим аспектом є міжнародне співробітництво. Враховуючи глобальний характер Інтернету, забезпечення сумісності криптографічних рішень у різних країнах стає дедалі складнішим через відмінності у правових системах та інтересах. Стандартизація алгоритмів шифрування, погоджена на міжнародному рівні, залишається критично важливою, але важко досяжною метою.

1.3 Порівняння шифрів Хілла, Віженера, Бофорта

1.3.1 Характеристика шифру Хілла

Спочатку вибирається m - буквеного алфавіту й кодуємо кожну букву одним числом із множини $\{0,1, 2,..., 1\ m - \}$. Нехай алфавіт містить звичайні 33 літери алфавіту української мови (тобто $m = 33$), як зображено на рис. 1.1

А	Б	В	…	Ю	Я
0	1	2	…	31	32

Рисунок 1.1 - Алфавіт української мови

Тепер будемо ставити у відповідність кожній біграмі (α , β) таке число P = 33α + β ∈ { 0,1,...,33^2 -1} .

Наприклад, біграмі «HI» є відповідним ціле натуральне число P = 33*17+11=572 . Далі, зручно мати прості правила або функції шифрування й розшифрування. Ці функції повинні здійснювати ін'єктивне відображення, яке зображено на рис. 1.2

$$P \xrightarrow{f} C \xrightarrow{f^{-1}} P,$$

Рисунок 1.2 – Шифрування та розшифрування

де P - множина відкритих повідомлень (у цьому випадку біграм), але вже у формі чисел із множини Z/33^2Z={0,1,...,33^2 -1 }. Функція шифрування повинна здійснювати перестановку на множині Z/33^2Z. Вибір відображення f , що шифрує, найпростіше взяти у вигляді афінного перетворення, що показано на рис. 1.3

$$C = aP + b \left(\mathrm{mod}\, N^2\right).$$

Рисунок 1.3 – Формула шифрування

Тут a , b - цілі числа, N = 33 .

Для того, щоб існувало обернене відображення f^{-1} , необхідно вимагати, щоб (a, N^2)=1. Тобто, a і N^2 – взаємно прості, що відображається формулою[15]:

$$P = f^{-1}(C) = a^{-1}C - a^{-1}b \left(\mathrm{mod}\, N^2\right). \qquad (1.1)$$

На рисунку 1.4 зображений алгоритм Л.Хілла

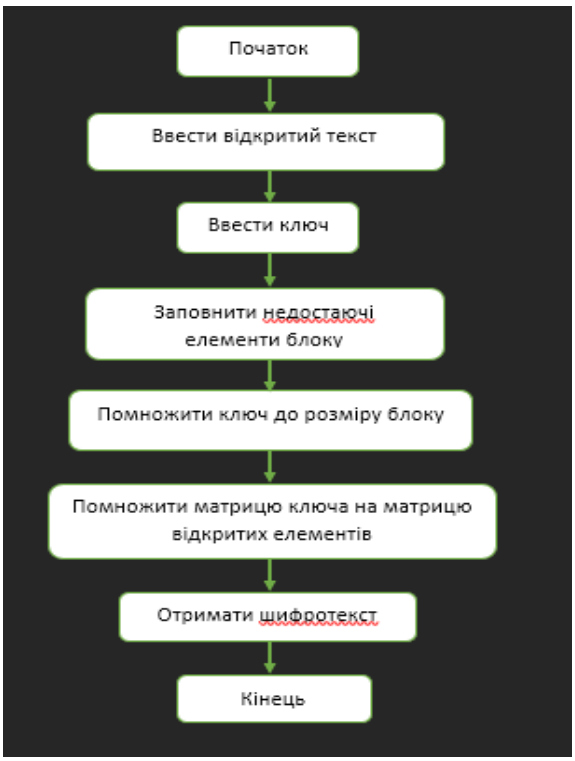

Рисунок 1.4 - Алгоритм Л.Хілла

1.3.2 Характеристика шифру Віженера

Однією із старих і найбільш відомих багатоалфавітних криптосистем є система Віженера, названа на честь французького криптографа Блейза Віженера (Vigenere), відома також як шифр Трітеміуса. Цей метод був вперше опублікований в 1586 році. У даному шифрі ключ задається набором з d літер.

Такі набори підписуються із повторенням під повідомленням, а, потім, отриману послідовність складають із відкритим текстом по модулю n (потужність алфавіту). Тобто виходить наступна формула:

$$Vig_d(m_i)=(m_i+k_{i \bmod d})(\bmod n) \qquad (1.2)$$

Літеру шифротексту можна знаходити також із таблиці (див. додаток А), як перетин стовпця, визначуваного літерою відкритого тексту, і рядка, визначуваною літерою ключа.

В окремому випадку, при d=1, отримуємо шифр Цезаря[15].

На рисунку 1.5 зображений алгоритм Б.Віженера

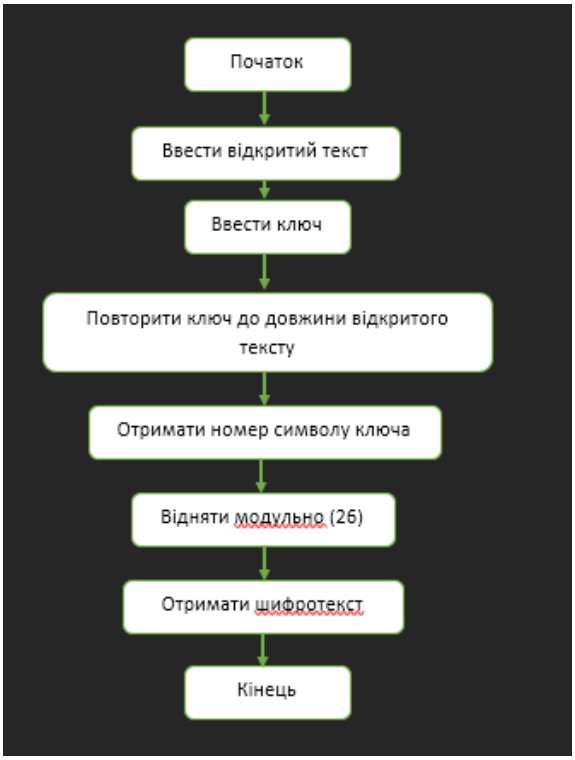

Рисунок 1.5 - Алгоритм Б.Віженера

1.3.3 Характеристика шифру Бофорта

Іншим різновидом шифру Віженера є шифр Бофорта (Beaufort, в деякій літературі читається як Бьюфорт), названий на честь адмірала сера Френсіса Бофорта - творця шкали для визначення швидкості вітру.

Його рядками є рядки квадрата Віженера, записані в зворотному порядку, а перший і останній рядки поміняні місцями (див. додаток Б).

Формула перетворення буде наступна:

$$Bof_d(m_i)=(k_{i \bmod d}-m_i)(\text{mod } n) \quad (1.3)$$

В обох випадках зворотна підстановка легко визначається з квадрату, або за формулами:

$$Vig_d^{-1}(m_i)=(m_i-k_{i \bmod d})(\text{mod } n) \quad (1.4)$$

$$Bof_d^{-1}(m_i)=Bof_d(m_i)=(k_i-m_{i \bmod d})(\text{mod } n) \quad (1.5)$$

Повторне застосування двох або більшої кількості шифрів Віженера називатиметься *складеним шифром Віженера*. Він визначається рівнянням:

$$Vig^*(m_i)=(m_i + k_{i \bmod dk} + l_{i \bmod dl} +...+ s_{i \bmod ds})'(\text{mod } n) \quad (1.6)$$

де $k_i + l_i + ... + s_i$, взагалі кажучи, мають різні періоди dk, dl, ..., ds відповідно. Період їх суми $k_i + l_i + ... + s_i$ буде найменшим спільним кратним окремих періодів.

Якщо ключ k не повторюється, то вийде *шифр Вернама*. Якщо в якості ключа використовується текст, що має смисл, то маємо шифр "біжучого ключа"[15].

На рисунку 1.6 зображений алгоритм Ф.Бофорта

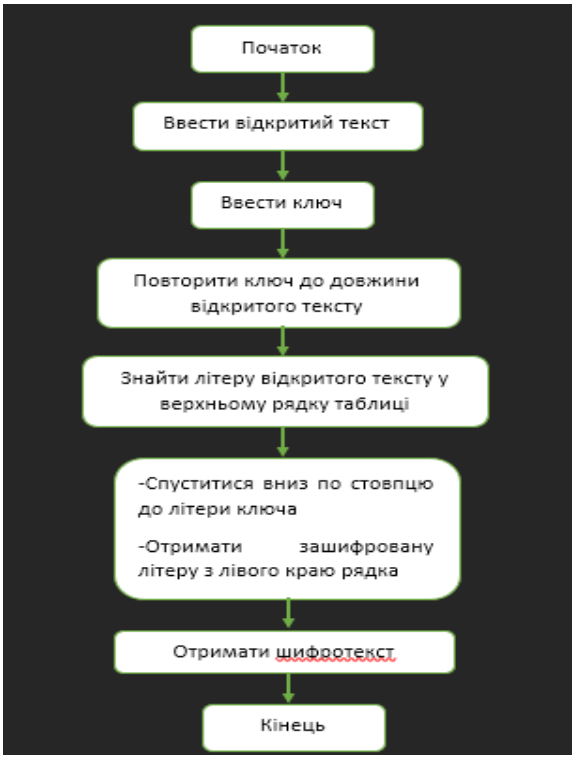

Рисунок 1.6 - Алгоритм Ф.Бофорта

Тобто базуючись на цих дослідженнях маємо такі результати:

Шифр Віженера — простий та історично популярний, але піддається криптоаналізу (аналіз повторів).

Шифр Бофорта — схожий на Віженера, але навпаки — інверсна таблиця.

Колись використовувався у британському флоті.

Шифр Хілла — найцікавіший математично, бо працює з лінійною алгеброю, але складніший в реалізації та вразливий до атак, якщо ключ-матриця погано обрана.

1.4 Вимоги до алгоритмів шифрування

Алгоритми шифрування повинні відповідати ряду вимог, щоб забезпечити надійний захист інформації в сучасних умовах[24]. Перш за все, важливою вимогою є конфіденційність — алгоритм має гарантувати, що сторонні особи не зможуть отримати доступ до зашифрованих даних без відповідного ключа. Це досягається за рахунок сильних криптографічних перетворень та використання ключів достатньої довжини (не менше 128 біт для симетричних алгоритмів). Крім того, алгоритм повинен бути стійким до всіх відомих типів криптоаналізу, таких як перебір ключів (brute-force), аналіз за частотою, атаки з вибраним відкритим текстом, атаки через побічні канали, а також мати потенційну стійкість до квантових атак, які можуть виникнути в майбутньому. Зворотність є ще однією критичною властивістю — можливість точно відновити початкові дані з шифрованих за допомогою правильного ключа. Алгоритм також повинен бути ефективним, тобто працювати з високою швидкістю навіть при обробці великих обсягів інформації, і бути придатним до використання на пристроях з обмеженими ресурсами, наприклад, мобільних телефонах чи пристроях Інтернету речей (IoT).

Надзвичайно важливим є надійне управління ключами: їх безпечне зберігання, контроль доступу, можливість оновлення або відкликання, а також підтримка безпечного обміну ключами, особливо в асиметричних системах шифрування. Крім того, сучасні алгоритми повинні бути гнучкими та адаптивними — тобто легко впроваджуватись у різні програмні та апаратні платформи, а також масштабуватись у разі потреби, наприклад, через зміну довжини ключа. Важливою

вимогою є також їх стійкість у часі: вони повинні залишатися безпечними протягом тривалого періоду, з урахуванням розвитку обчислювальних потужностей та появи нових загроз. Надійні алгоритми, як правило, стандартизовані авторитетними міжнародними організаціями, такими як ISO чи NIST, що гарантує публічну експертизу, відкритість і відповідність сучасним вимогам безпеки[14], [12]. Алгоритм шифрування має бути широко перевіреним спільнотою криптографів, пройти незалежний аудит, тестування та практичне застосування. І нарешті, в алгоритмах не повинно бути жодних бекдорів — прихованих механізмів доступу, які дозволяють стороннім особам, включно з урядовими структурами, отримати несанкціонований доступ до інформації.

Таким чином, ефективний алгоритм шифрування повинен поєднувати в собі надійність, продуктивність, гнучкість і відкритість, забезпечуючи високий рівень захисту даних у цифровому середовищі.

Висновки до першого розділу

У першому розділі кваліфікаційної роботи було розглянуто ключові принципи шифрування, які лежать в основі сучасних систем захисту інформації — конфіденційність, цілісність, автентифікація та контроль доступу. Ці поняття формують базу для надійного зберігання та передавання даних у цифрову епоху.

Також охарактеризовано основні типи шифрування — симетричне (наприклад, AES) та асиметричне (RSA), кожне з яких має свої переваги та недоліки залежно від контексту застосування. Розглянуто принципи роботи алгоритмів декодування, які дозволяють уповноваженим користувачам безпечно отримувати доступ до захищеної інформації. Особливу увагу приділено сучасним викликам, таким як потреба в стійкості до квантових атак, впровадження гомоморфного шифрування та використання криптографії в технологіях блокчейн. Водночас, були проаналізовані суперечливі аспекти, пов’язані з урядовим контролем, бекдорами та

міжнародним співробітництвом. Крім того, розглянуто та порівняно три класичні шифри — Хілла, Віженера та Бофорта. Шифр Хілла демонструє застосування лінійної алгебри та матриць у криптографії, в той час як шифри Віженера і Бофорта є багатоалфавітними підстановками з циклічним використанням ключа. Попри простоту реалізації, ці алгоритми сьогодні мають здебільшого навчальне або історичне значення, поступаючись сучасним криптосистемам у безпеці та ефективності. Таким чином, Розділ 1 формує теоретичну основу для розуміння як традиційних, так і сучасних підходів до шифрування, що є критично важливим для розробки ефективних програм захисту даних у сучасному інформаційному середовищі.

Підсумовуючи, алгоритми шифрування і декодування відіграють ключову роль у забезпеченні захисту конфіденційної інформації. Вони дозволяють уповноваженим користувачам безпечно отримувати доступ до даних, захищаючи їх від несанкціонованого доступу. Завдяки використанню сучасних криптографічних рішень, впровадженню багатофакторної аутентифікації та дотриманню актуальних протоколів безпеки, як окремі користувачі, так і організації можуть значно зміцнити інформаційну безпеку. Проте, попри технічний прогрес, етичні, юридичні та політичні аспекти залишаються невід’ємною частиною дискурсу навколо шифрування в сучасному цифровому світі.

РОЗДІЛ 2 ОГЛЯД ПРОГРАМ ТА СЕРЕДОВИЩ, ЯКІ ВИКОРИСТОВУЮТЬСЯ ДЛЯ ШИФРУВАННЯ ДАНИХ

2.1 Програми шифрування файлів і папок

Encrypting File System (EFS) — система шифрування даних, що реалізує шифрування на рівні файлів в операційних системах Microsoft Windows NT (починаючи з Windows 2000 і вище), за винятком «домашніх» версій (Windows XP Home Edition, Windows Vista Basic, Windows Vista Home Premium, Windows 7 Starter (Home Basic і Premium), Windows 10 Pro, Enterprise, and Education editions, Windows Server 2016, Windows Server 2019. Дана система надає можливість прозорого шифрування даних[en], що зберігаються у розділах з файловою системою NTFS, для захисту потенційно конфіденційних даних від несанкціонованого доступу за наявності фізичного доступу сторонніх осіб до комп'ютера і дисків. Аутентифікація користувача та права доступу до ресурсів, що мають місце в NT, працюють, коли операційна система завантажена, але при фізичному доступі до системи з'являється можливість завантажити іншу ОС, та обійти ці обмеження. EFS використовує симетричне шифрування для захисту файлів, а також шифрування, засноване на парі відкритий/закритий ключ, для захисту випадково згенерованого ключа шифрування для кожного файлу. За замовчуванням закритий ключ користувача захищений за допомогою шифрування користувальницьким паролем, і захищеність даних залежить від стійкості пароля користувача[3].

До таких програм відносяться:

1. VeraCrypt – потужна програма для створення зашифрованих контейнерів та повного шифрування дисків[5].
2. BitLocker – вбудоване в Windows рішення для шифрування дисків[6].
3. FileVault – аналог BitLocker для macOS. [18]
4. AxCrypt – простий у використанні інструмент для шифрування файлів. [19]

5. 7-Zip (з шифруванням AES-256) – дозволяє створювати зашифровані архіви[20], головне меню програми зображено на рис. 2.1

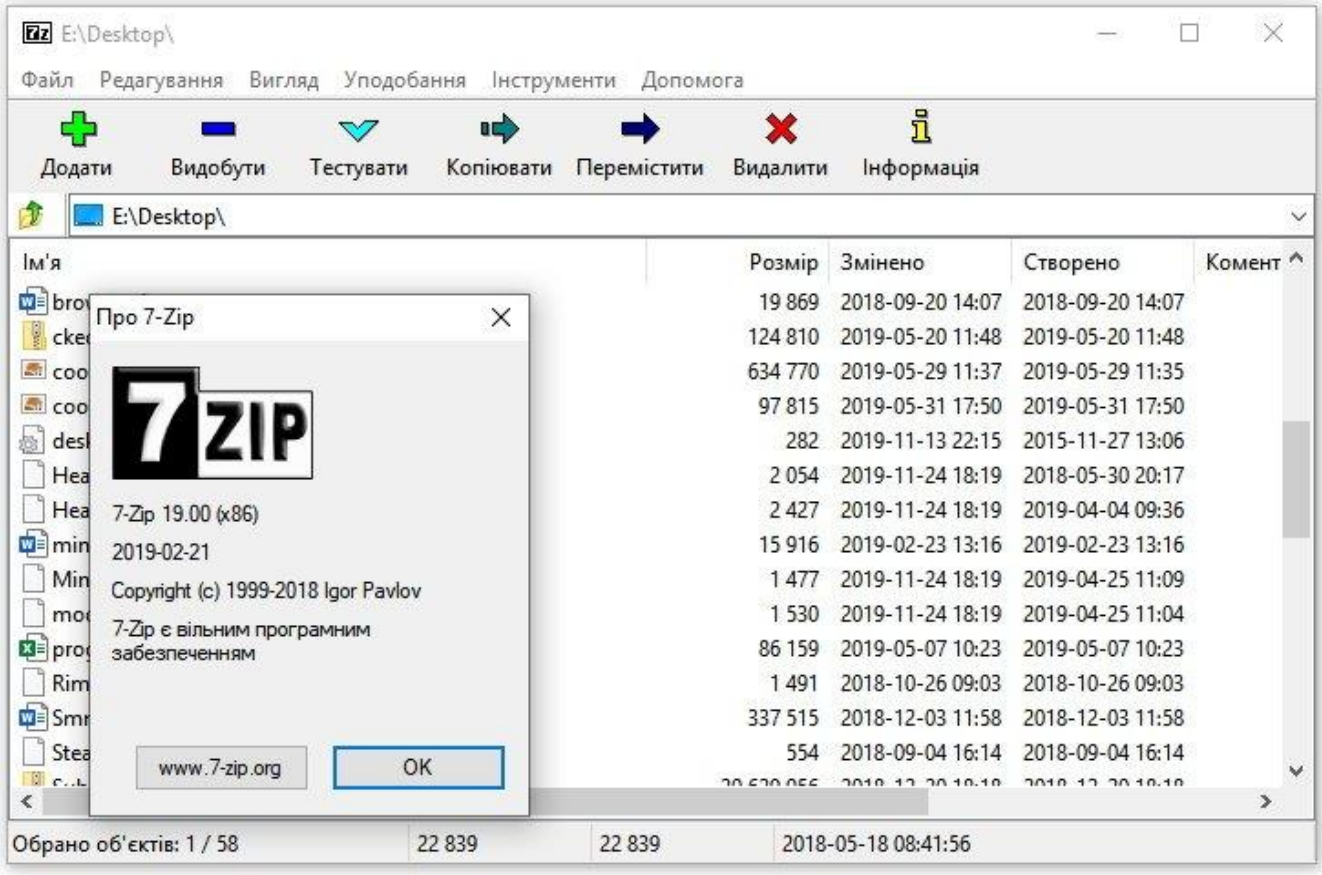

Рисунок 2.1 - Головне меню програми -7-Zip

Одним із найпоширеніших способів використання шифрування є захист файлів і дисків, що дозволяє користувачам зберігати свої дані в безпечному вигляді. Серед популярних програм для цієї мети можна виділити VeraCrypt, BitLocker та FileVault. VeraCrypt є потужним інструментом для шифрування, який дозволяє створювати зашифровані контейнери – файли, які можуть містити інші файли, а також підтримує шифрування цілих розділів або дисків. Ця програма є відкритим програмним забезпеченням і широко використовується для захисту чутливих даних. BitLocker, що вбудований у операційну систему Windows, надає можливість шифрувати системні та знімні диски, використовуючи алгоритми AES-128 і AES-

256. Ця технологія працює у фоновому режимі, забезпечуючи прозору для користувача безпеку. FileVault виконує аналогічну функцію для користувачів macOS, дозволяючи шифрувати весь диск за допомогою алгоритму XTS-AES-128. Це рішення забезпечує високий рівень безпеки та інтегрується в систему без потреби встановлення додаткового програмного забезпечення. Крім цього, існують інші програми, такі як AxCrypt, 7-Zip та WinRAR, які надають можливість зашифровувати окремі файли чи архіви, використовуючи AES-256, що є одним із найнадійніших алгоритмів шифрування.

2.2 Огляд програм шифрування повідомлень та електронної пошти

Одним із ключових аспектів кібербезпеки є захист повідомлень, що передаються через електронну пошту або месенджери. Для цього використовуються різні технології серед них можна виділити:

1.PGP є однією з найпоширеніших систем шифрування для захисту електронних повідомлень[21]. Система використовує асиметричне шифрування, де для шифрування даних застосовується відкритий ключ одержувача, а для їх розшифровки — його закритий ключ. Це дозволяє гарантувати, що навіть якщо зловмисник перехопить зашифроване повідомлення, без закритого ключа він не зможе його прочитати. Відкритий аналог PGP — GnuPG (або GPG) — є безкоштовним і відкритим програмним забезпеченням, що забезпечує ті самі функції шифрування, але доступне для більш широкої аудиторії, включаючи розробників і користувачів з відкритим кодом[22].

2.ProtonMail — це захищений сервіс електронної пошти, який надає можливість шифрування листів без необхідності використовувати додаткові інструменти. ProtonMail підтримує PGP-шифрування за замовчуванням, що дозволяє користувачам обмінюватися зашифрованими електронними листами як всередині екосистеми ProtonMail, так і з користувачами зовнішніх поштових

сервісів за допомогою пароля. ProtonMail використовує наскрізне шифрування, що означає, що тільки відправник і одержувач можуть прочитати зміст листа, що значно підвищує рівень безпеки і конфіденційності[9].

3.Signal — це один з найбезпечніших месенджерів, який забезпечує наскрізне шифрування для всіх повідомлень і дзвінків. Використовує Signal Protocol, що є стандартом для наскрізного шифрування в сучасних месенджерах. Signal гарантує, що тільки відправник і отримувач повідомлення можуть розшифрувати його зміст. Цей протокол забезпечує захист від перехоплення і дозволяє досягти високого рівня конфіденційності даних, що передаються[8].

4.Telegram — популярний месенджер, що також підтримує наскрізне шифрування в рамках секретних чатів. Секретні чати Telegram використовують власний протокол шифрування, що дозволяє забезпечити конфіденційність переданих повідомлень. Вони працюють за принципом "ключа на пристрої", що гарантує відсутність можливості для сторонніх осіб, у тому числі адміністраторів сервісу, прочитати повідомлення[23].

Порівняння PGP, GPG і сучасних месенджерів:

Таблиця 2.1Порівняння PGP, GPG і сучасних месенджерів

Характеристика	PGP / GPG	ProtonMail	Signal	Telegram
Тип сервісу	Шифрування електронної пошти	Електронна пошта	Месенджер	Месенджер
Тип шифрування	Асиметричне (RSA, ECC)	Вбудований PGP (асиметричне)	Наскрізне (End-to-End)	Наскрізне тільки в секретних чатах

Продовження таблиці 2.1

Наскрізне шифрування	Ні (вимагає ручного налаштування)	Так (автоматично між користувачами ProtonMail)	Так (за замовчуванням)	Тільки в секретних чатах (не за замовчуванням)
Простота використання	Низька (потрібні технічні знання)	Висока (інтерфейс дружній до користувача)	Висока	Висока
Використання в браузері	Можливе через плагіни	Так (веб-інтерфейс з шифруванням)	Ні (мобільний/десктоп клієнт)	Так
Відкритий вихідний код	Так	Частково (деякі компоненти закриті)	Так	Частково
Доступність бекапу ключів	Користувач зберігає ключі самостійно	Автоматизовано (але з обмеженнями)	Так, через PIN-код і резервні фрази	Частково, залежить від типу чату
Цільове використання	Безпечна електронна пошта	Проста безпечна пошта	Безпечні дзвінки та повідомлення	Універсальний чат з опціями шифрування

PGP / GPG: Використовуються для шифрування електронної пошти, застосовують асиметричне шифрування, що дозволяє забезпечити конфіденційність в обміні повідомленнями.

ProtonMail: Сервіс електронної пошти з вбудованим шифруванням за допомогою PGP, що забезпечує простоту у використанні і високий рівень безпеки.

Signal: Месенджер з наскрізним шифруванням, що забезпечує конфіденційність для особистих повідомлень та дзвінків.

Telegram: Підтримує наскрізне шифрування в секретних чатах, що дає можливість зберігати конфіденційність повідомлень, що передаються.

Ці технології є важливими інструментами для забезпечення безпеки та конфіденційності особистої інформації в цифровому середовищі. Вибір між ними залежить від того, що саме потрібно захистити: електронну пошту, месенджери чи приватне спілкування в Інтернеті.

2.3 Огляд програм шифрування трафіку та VPN

Щоб захистити інформацію під час використання інтернету, широко застосовуються VPN (віртуальні приватні мережі) та анонімні мережі. OpenVPN та WireGuard є двома найпопулярнішими VPN-протоколами. OpenVPN пропонує високий рівень безпеки та широко використовується у корпоративних мережах. WireGuard, у свою чергу, є більш швидким і легким у налаштуванні рішенням, що забезпечує надійне шифрування з меншою затримкою у з'єднанні. Крім VPN, для анонімного перегляду веб-сторінок використовується мережа Tor[16]. Вона дозволяє користувачам приховувати свою IP-адресу та маршрутизувати трафік через кілька серверів, що забезпечує високий рівень конфіденційності.

Характеристики програм шифру трафіку та VPN наступні:

1.OpenVPN є одним з найпопулярніших VPN-протоколів, який використовує для шифрування високий рівень безпеки. Він активно застосовується в

корпоративних мережах і надає гнучкість у налаштуваннях, підтримуючи різні методи аутентифікації та алгоритми шифрування. Використовує технології SSL/TLS, що дозволяє створити захищений тунель для передавання даних між клієнтом і сервером. Завдяки великій спільноті користувачів і надійній підтримці OpenVPN залишається одним з основних виборів для забезпечення безпечного з'єднання в мережі Інтернет[17].

2.WireGuard — це новий, швидкий і легкий VPN-протокол, який пропонує високу продуктивність при меншій затримці зв'язку. Він набирає популярності завдяки своїй простоті в налаштуванні та використанні. WireGuard використовує сучасні методи шифрування, такі як Curve25519, ChaCha20, Poly1305 і BLAKE2, що забезпечує високу швидкість передачі даних при одночасному збереженні рівня безпеки. Оскільки протокол спрощує налаштування і зменшує кількість необхідних компонентів, він може бути ідеальним вибором для тих, хто шукає швидкий та ефективний спосіб захисту свого інтернет-з'єднання[10].

3.Мережа Tor — це система для анонімного перегляду веб-сайтів і захисту конфіденційності в Інтернеті. Вона працює шляхом маршрутизації трафіку через кілька рівнів серверів, що надають користувачам можливість приховувати свою реальну IP-адресу і здійснювати анонімний серфінг в мережі. Tor використовує принцип «цибулевої» маршрутизації, де кожен переданий пакет зашифровується кілька разів, і кожен сервер в ланцюгу може бачити лише попередній і наступний вузол, що ускладнює відстеження користувача. Мережа Tor особливо популярна серед тих, хто прагне зберегти свою анонімність під час перегляду веб-сторінок або доступу до заблокованих ресурсів[16].

Порівняння OpenVPN, WireGuard та Tor:

Таблиця 2.2 порівняння OpenVPN, WireGuard та Tor

Характеристика	OpenVPN	WireGuard	Tor
Тип технології	VPN-протокол	VPN-протокол	Анонімна мережа (анонімайзер)

Продовження таблиці 2.2

Шифрування	AES-256, RSA, TLS	ChaCha20, Poly1305	Шифрування трафіку у 3 шарах (мульти-хоп)
Швидкість роботи	Середня	Висока (дуже швидкий)	Низька (через кілька вузлів маршрутизації)
Анонімність	Немає повної анонімності	Немає повної анонімності	Висока (при правильному використанні)
Призначення	Захист трафіку, доступ до VPN-серверів	Захист трафіку, сучасна заміна OpenVPN	Анонімний доступ до Інтернету, приховані сервіси
Налаштування	Гнучке, але складне	Просте (менше коду, легка конфігурація)	Просте для користувача, складне для розгортання
Підтримка платформ	Windows, macOS, Linux, Android, iOS	Windows, macOS, Linux, Android, iOS	Windows, macOS, Linux, Android
Використання в реальному часі	Підходить для потокового відео, ігор	Підходить для потокового відео, ігор	Непридатний для відео/ігор через низьку швидкість
Відкритий вихідний код	Так	Так	Так
Безпека	Висока (перевірений роками)	Висока (але ще не так широко перевірений)	Висока (при правильному використанні)

OpenVPN: висока безпека, гнучкість, велика спільнота, але може бути повільним через складну конфігурацію.
WireGuard: швидкість, простота налаштування, сучасне шифрування, менше навантаження на системи, але ще не має такої популярності, як OpenVPN.
-: високий рівень анонімності, приховування IP-адреси, але знижена швидкість через маршрутизацію через кілька серверів.
Усі ці інструменти використовуються для різних цілей: OpenVPN і WireGuard більше орієнтовані на безпечне з'єднання і захист трафіку, в той час як Tor ставить акцент на анонімність і конфіденційність в мережі.

2.4 Огляд програм шифрування баз даних та хмарних сховищ

У корпоративному середовищі велика увага приділяється захисту баз даних. Для цього використовуються технології шифрування, такі як Transparent Data Encryption (TDE), що доступна в Microsoft SQL Server та інших системах керування базами даних. TDE автоматично шифрує всі дані, що зберігаються у базі, забезпечуючи їх захист від зловмисників, навіть у випадку фізичного доступу до пристрою зберігання. У Windows існує можливість використовувати Encrypting File System (EFS), який дозволяє зашифровувати файли у файловій системі NTFS. Це корисна функція для тих, хто хоче забезпечити безпеку конфіденційних документів без потреби у додатковому програмному забезпеченні. Зашифровані EFS-файли можна відкрити лише користувачем, який їх шифрував, або уповноваженим адміністратором за умови збереження сертифікату шифрування.

Для захисту файлів у хмарних сховищах, таких як Google Drive, Dropbox або OneDrive, застосовується Cryptomator. Це програмне забезпечення дозволяє створювати зашифровані контейнери, у яких файли зберігаються у зашифрованому вигляді навіть у хмарному середовищі[7]. Таким чином, навіть якщо хмарний обліковий запис буде скомпрометований, файли залишаться недоступними без

пароля користувача. Такі рішення стають дедалі актуальнішими в умовах зростаючих загроз у сфері кібербезпеки. Вони дозволяють реалізовувати політики захисту даних відповідно до міжнародних стандартів (наприклад, ISO/IEC 27001) та нормативних вимог (GDPR, ЗУ «Про захист персональних даних») [7].

Шифрування даних у корпоративному середовищі: TDE, EFS та Cryptomator

У сучасному корпоративному середовищі забезпечення безпеки даних є важливою складовою кібербезпеки. Для цього використовуються різні технології шифрування, які допомагають захищати чутливу інформацію як на рівні баз даних, так і при зберіганні файлів у локальних чи хмарних сховищах. До таких рішень належать Transparent Data Encryption (TDE), Encrypting File System (EFS) та Cryptomator.

Характеристи цих рішень наступні:

1.TDE — це технологія шифрування, яка забезпечує автоматичне шифрування всіх даних у базі даних. Вона доступна в Microsoft SQL Server та інших системах керування базами даних. TDE шифрує не лише самі дані, але й журнал транзакцій, таким чином гарантується їх захист від несанкціонованого доступу. Це особливо важливо в умовах, коли існує ризик фізичного доступу до носіїв інформації або серверів. Завдяки TDE, навіть якщо зловмисник отримає доступ до файлів бази даних, він не зможе прочитати дані без відповідного ключа шифрування.

2.EFS — це вбудована функція в операційних системах Windows, яка дозволяє шифрувати окремі файли або папки в файловій системі NTFS. Це корисний інструмент для захисту конфіденційних документів без необхідності використовувати стороннє програмне забезпечення. Шифрування через EFS дозволяє створювати файли, доступні лише користувачам, які мають відповідний сертифікат шифрування. У разі втрати чи компрометації даного сертифіката, доступ до зашифрованих файлів може отримати тільки уповноважений адміністратор, що додає додатковий рівень захисту.

3.Cryptomator — це програмне забезпечення для захисту файлів у хмарних сховищах, таких як Google Drive, Dropbox та OneDrive. Воно дозволяє створювати

зашифровані контейнери, в яких файли зберігаються у зашифрованому вигляді навіть в хмарному середовищі. Таким чином, навіть якщо зловмисник отримає доступ до вашого хмарного облікового запису, він не зможе прочитати вміст файлів без пароля, який використовується для шифрування. Cryptomator дозволяє користувачам захищати свої файли без додаткових складних налаштувань і інтегрується з популярними хмарними платформами.

Порівняння TDE, EFS та Cryptomator:

Таблиця 2.3 Порівняння TDE, EFS та Cryptomator

Характеристика	TDE (Transparent Data Encryption)	EFS (Encrypting File System)	Cryptomator
Призначення	Шифрування баз даних на рівні зберігання	Шифрування окремих файлів/папок у файловій системі	Шифрування файлів перед зберіганням у хмарі
Рівень шифрування	На рівні бази даних або таблиць	На рівні файлів і папок у NTFS	На рівні файлів та віртуального сховища (трезору)
Цільові користувачі	Адміністратори баз даних, підприємства	Користувачі Windows, які потребують локальне шифрування	Приватні користувачі, які використовують хмару
Платформи	SQL Server, Oracle, MySQL тощо	Windows NTFS	Windows, macOS, Linux, Android, iOS

Продовження таблиці 2.3

Прозорість для користувача	Так – працює автоматично на фоні	Так – інтегрується з ОС, прозоро при доступі	Ні – користувач повинен відкрити сховище вручну
Захист при крадіжці файлу	Так, шифрує файли БД на диску	Так, але лише на NTFS-дисках	Так, шифрує файли індивідуально і метадані
Керування ключами	Через систему управління ключами БД	Через сертифікати користувача Windows	Пароль користувача (локально, без сторонніх серверів)
Підтримка хмари	Немає прямої інтеграції	Обмежена (тільки якщо NTFS використовується в хмарі)	Повна – розроблений для роботи з хмарними сервісами
Відкритий код	Ні (залежить від реалізації)	Ні	Так (повністю open-source)
Простота використання	Вимагає налаштування адміністратором	Простий у використанні, але тільки в Windows	Зручний для звичайних користувачів
Призначення шифрування	Захист БД від викрадення фізичних файлів	Захист окремих локальних файлів і папок	Захист особистих файлів у хмарі від стороннього доступу

TDE: Шифрує дані в базах даних та їх журнали, захищаючи їх на рівні серверів і забезпечуючи захист від фізичного доступу до даних.

EFS: Пропонує шифрування окремих файлів у файловій системі NTFS, що дозволяє зберігати конфіденційні документи в безпеці навіть на локальних пристроях.

Cryptomator: Зосереджений на шифруванні файлів у хмарних сховищах, забезпечуючи додатковий рівень захисту для даних, що зберігаються в хмарах.

2.5 Аналіз недоліків та переваг середовищ шифрів даних

Середовища шифрування мають ряд особливостей. Розглянуто аналізу особливостей які надано в табл. 2.4

Таблиця 2.4 - Аналіз недоліків та переваг

Категорія	Назва програми/технології	Переваги	Недоліки
Шифрування файлів і дисків	VeraCrypt	Відкрите ПЗ, підтримка контейнерів і повного шифрування дисків	Потребує налаштування, не інтегроване в ОС
	BitLocker	Інтеграція в Windows, прозоре фонове шифрування, підтримка TPM	Доступний лише в деяких версіях Windows
	FileVault	Інтеграція в macOS, просте шифрування всього диска	Лише для macOS, обмежена конфігурація
	AxCrypt	Зручний для окремих файлів, просте ПЗ для користувача	Обмежений функціонал, платні функції
	7-Zip (AES-256)	Безкоштовне, сильне шифрування архівів	Не підходить для повного шифрування диска

Продовження таблиці 2.4

Шифрування повідомлень	PGP / GnuPG (GPG)	Асиметричне шифрування, високий рівень безпеки	Складність у використанні, потрібне управління ключами
	ProtonMail	Вбудоване шифрування листів, легкість використання	Обмеження у взаємодії з іншими поштовими сервісами
	Signal	Наскрізне шифрування, відкритий код, висока конфіденційність	Потрібна інсталяція додатку, не всі функції в десктоп-версії
	Telegram (секретні чати)	Прості у використанні, доступне наскрізне шифрування	Шифрування тільки в секретних чатах, не за замовчуванням

Продовження таблиці 2.4

Шифрування трафіку / VPN	OpenVPN	Гнучкість, надійне шифрування, підходить для корпоративних мереж	Складніше налаштування порівняно з іншими VPN
	WireGuard	Висока швидкість, легке налаштування, сучасний протокол	Новий стандарт, ще не всюди підтримується
	Tor	Анонімність, маршрутизація через вузли, висока конфіденційність	Повільне з'єднання, блокування деяких сайтів
Шифрування БД і хмарних файлів	TDE (SQL Server)	Прозоре шифрування БД, інтеграція в СУБД	Потребує професійних/комерційних версій, не шифрує резервні копії
	EFS (Windows)	Інтеграція в ОС, зручно для користувача, не потребує сторонніх програм	Обмеження NTFS, залежність від облікового запису
	Cryptomator	Вільне ПЗ, просте рішення для хмарних файлів (Google Drive тощо)	Призначене для приватного використання, не підходить для великих компаній

Отже аналізуючи недоліки і переваги середовищ, обрані позиції для реалізації у власному програмному додатку RPP

Висновки до другого розділу

У процесі дослідження програмного забезпечення для шифрування даних було проаналізовано широкий спектр сучасних інструментів, призначених для захисту файлів і папок, повідомлень, мережевого трафіку, баз даних та хмарних сховищ. Зокрема, розглянуто функціональні можливості таких рішень, як VeraCrypt, AxCrypt, BitLocker, Signal, ProtonMail, OpenVPN, MySQL, MongoDB, MEGA тощо. Ці програми відзначаються високим рівнем криптографічного захисту, підтримкою актуальних стандартів шифрування та широкою сумісністю з різними операційними системами.Отримані результати свідчать про те, що шифрування залишається одним з найефективніших способів забезпечення інформаційної безпеки. Вибір відповідного програмного забезпечення дозволяє надійно захищати конфіденційні дані від несанкціонованого доступу, як у приватному, так і в корпоративному середовищі. Водночас ефективність таких засобів значною мірою залежить від правильного налаштування, регулярного оновлення та дотримання користувачами базових принципів безпеки — зокрема, використання складних паролів і багатофакторної автентифікації.

Таким чином, результати дослідження підтверджують доцільність і необхідність застосування спеціалізованого програмного забезпечення для шифрування в умовах зростання загроз кібербезпеці та постійного розширення цифрового простору. Ці технології сприяють дотриманню стандартів кібербезпеки, таких як ISO/IEC 27001, а також нормативних вимог, як GDPR (Загальний регламент захисту даних) або ЗУ «Про захист персональних даних», дозволяючи організаціям реалізувати належну політику захисту даних і забезпечити їх цілісність та

конфіденційність. В умовах зростаючих загроз кібербезпеці такі рішення стають надзвичайно актуальними для захисту конфіденційної інформації.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО СЕРЕДОВИЩА RPP

3.1 Характеристика шифру обраного для реалізації в середовищі RPP

Шифр Бофорта — це класичний поліалфавітний симетричний шифр, запропонований британським офіцером та гідрографом Френсісом Бофортом у XIX столітті. Він є модифікацією шифру Віженера, однак на відміну від останнього, шифрування здійснюється за допомогою віднімання значення символу відкритого тексту з відповідного символу ключа, а не додавання. Всі операції виконуються в межах модульної арифметики за модулем, який відповідає розміру алфавіту (наприклад, 26 для англійського алфавіту).

У шифрі Бофорта використовується спеціальна таблиця — таблиця Бофорта, яка є модифікованою версією таблиці Віженера з інверсією алфавітного порядку. Шифрування кожної літери полягає у пошуку в таблиці рядка, що відповідає символу ключа, і стовпця, в якому зустрічається літера відкритого тексту. В точці перетину знаходиться шифротекст.

Процес дешифрування ідентичний процесу шифрування, що забезпечується симетричністю алгоритму: той самий ключ і ті самі операції дозволяють відновити початковий текст. Це суттєво спрощує реалізацію шифру в програмному середовищі.

Шифр Бофорта демонструє хорошу стійкість до простих частотних атак, завдяки поліалфавітній природі, проте при використанні коротких або повторюваних ключів може бути вразливим до криптоаналізу методом Касіскі або частотного аналізу. Незважаючи на це, шифр має історичне значення і використовується як навчальний приклад реалізації поліалфавітних методів шифрування. Завдяки простоті свого алгоритму, він є зручним для реалізації у програмному середовищі та аналізу принципів симетричного шифрування.

3.2 Характеристика інструментів розробки середовища RPP

Python — це високорівнева, інтерпретована, об'єктно-орієнтована мова програмування з відкритим вихідним кодом, яка є однією з найпоширеніших у сучасному програмному середовищі. Її популярність зумовлена низкою переваг: простий і лаконічний синтаксис, велика спільнота розробників, багатий набір стандартних та сторонніх бібліотек, а також широкі можливості інтеграції з іншими мовами та технологіями.

У контексті реалізації середовища для шифрування (RPP) Python обрано як основну мову розробки з кількох причин:

Простота та читабельність коду. Це дозволяє зосередитися на реалізації логіки криптографічних алгоритмів, а не на синтаксичних деталях.

- Підтримка численних криптографічних бібліотек, таких як cryptography, pycryptodome, hashlib, base64, rsa, які забезпечують реалізацію як симетричних, так і асиметричних алгоритмів шифрування, генерацію ключів, хешування тощо.
- Потужні засоби для роботи з текстовими та двійковими даними. Це особливо важливо в контексті реалізації та обробки шифрованих повідомлень, файлів або мережевого трафіку.
- Гнучкість парадигм програмування. Python підтримує процедурне, об'єктно-орієнтоване та функціональне програмування, що дозволяє обрати найбільш відповідну модель для кожного компонента системи.

Крім того, Python добре підходить для створення графічних інтерфейсів користувача (GUI) завдяки таким бібліотекам, як Tkinter, PyQt або Kivy. Це дозволяє реалізувати зручне та інтуїтивне середовище для взаємодії користувача з криптографічними функціями програми. Також важливою є підтримка системного програмування, що дозволяє працювати з файлами, каталогами, мережевими з'єднаннями, а також взаємодіяти з операційною системою на низькому рівні. Нижче показаний фрагмент коду на рис. 3.1

```
def decrypt_message():
    encrypt_message()
root = tk.Tk()
root.title("RPP")
```

Рисунок 3.1 - Фрагмент коду в Python

Для реалізації програмного забезпечення на мові Python було використано середовище розробки Visual Studio Code (VS Code) — популярне кросплатформене IDE, розроблене компанією Microsoft. Це середовище поєднує в собі легкість текстового редактора з функціональністю повноцінної IDE.

До ключових переваг VS Code у контексті розробки криптографічного середовища належать:

- Розширюваність**.** Середовище має тисячі розширень, серед яких є специфічні для Python (Python, Pylance, Jupyter), а також для роботи з Git, середовищами виконання, тестування, налагодження.
- Інтелектуальне автодоповнення коду (IntelliSense). Завдяки цьому значно підвищується швидкість розробки та зменшується кількість помилок.
- Інтегроване налагодження. Дозволяє зручно відслідковувати логіку виконання програм, встановлювати точки зупинки (breakpoints), переглядати значення змінних у реальному часі, тощо.
- Вбудований термінал. Дозволяє запускати скрипти безпосередньо в IDE, створювати віртуальні середовища (venv, conda) та керувати залежностями проєкту.
- Контроль версій. VS Code має вбудовану інтеграцію з системою Git, що дозволяє ефективно керувати історією змін проєкту, співпрацювати над кодом у команді та створювати резервні копії.
- Підтримка багатьох мов та форматів. Це корисно, якщо середовище RPP включає файли конфігурації (наприклад, JSON, YAML) або документацію (Markdown, LaTeX).

Крім того, VS Code підтримує роботу з віддаленими серверами, Docker-контейнерами, а також має розширення для хмарної розробки, що робить його універсальним інструментом для проєктів різного рівня складності.

Діаграма кінцевого афтомату середовища RPP представлена на рис. 3.2

Користувач взаємодіє з графічним інтерфейсом програми, вводячи повідомлення, яке потрібно зашифрувати або розшифрувати. Спочатку користувач вводить сам текст повідомлення, а потім ключ для шифрування або дешифрування. Далі він обирає одну з двох дій — натискає кнопку «Зашифрувати» або «Розшифрувати».

У випадку натискання кнопки «Зашифрувати» викликається функція encrypt_message, а у випадку натискання кнопки «Розшифрувати» — функція decrypt_message.

Після цього відбувається обробка введених даних: отримується текст повідомлення та ключ. Далі викликається функція RPprocess(text, key), яка здійснює основну логіку обробки тексту.

У процесі обробки кожен символ повідомлення аналізується окремо. Якщо символ є літерою, виконуються обчислення значень, пов’язаних з текстом та ключем (ймовірно, позиції символів у алфавіті). Потім обчислюється зашифрований символ. Якщо символ не є літерою, він обробляється іншим чином (наприклад, додається до результату без змін).

Після завершення обробки всіх символів формується зашифрований (або розшифрований) рядок. Отриманий результат виводиться у графічному інтерфейсі користувача (GUI).

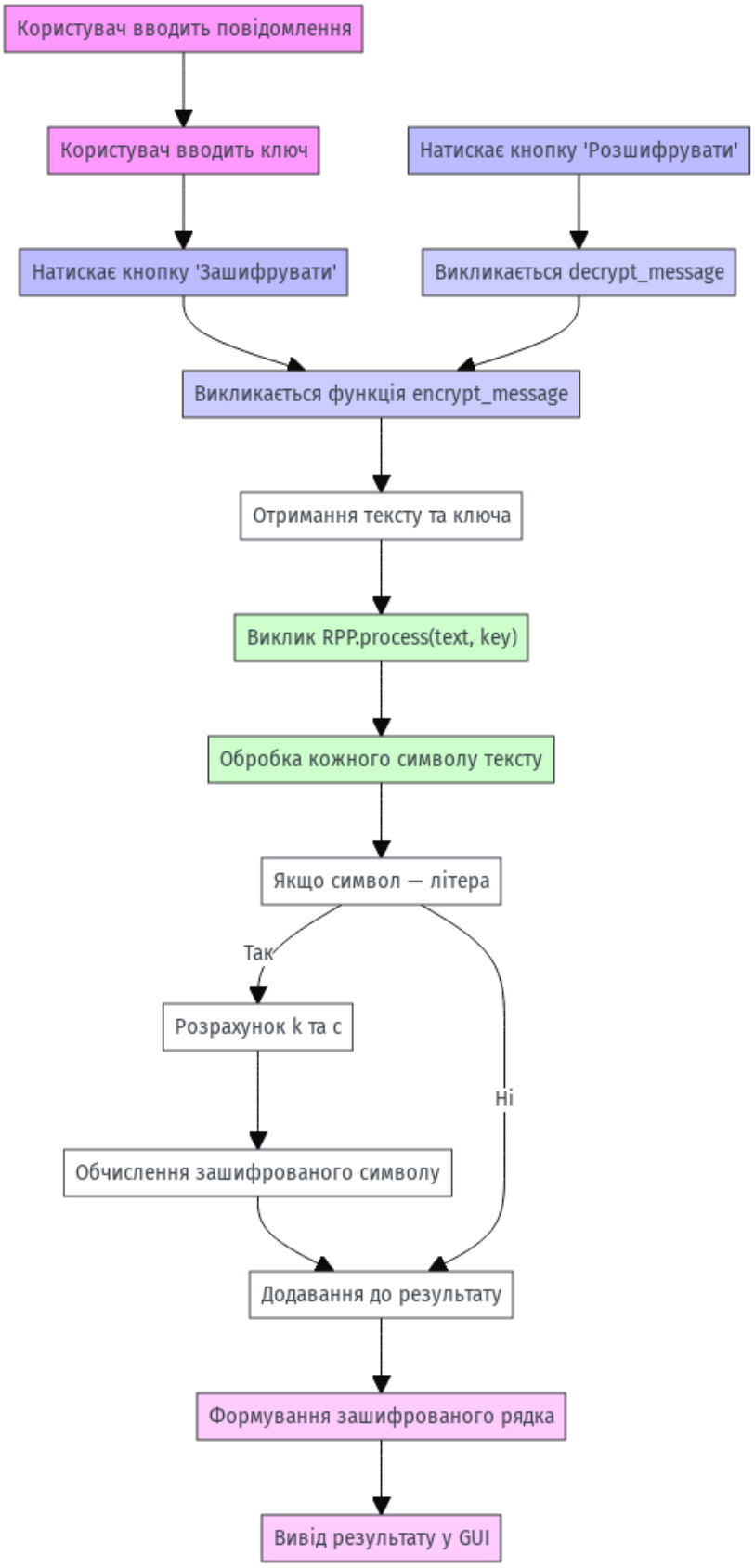

Рисунок 3.2 — Діаграма кінцевого автомату середовища RPP

3.3 Демонстрація тестової роботи середовища RPP

Щоб відкрити середовище RPP треба два рази натиснути на його ярлик на робочому столі, а потім вже у відкритому VScode натиснути кнопку Run python file. Відкриється вікно, зображене, на рис 3.1

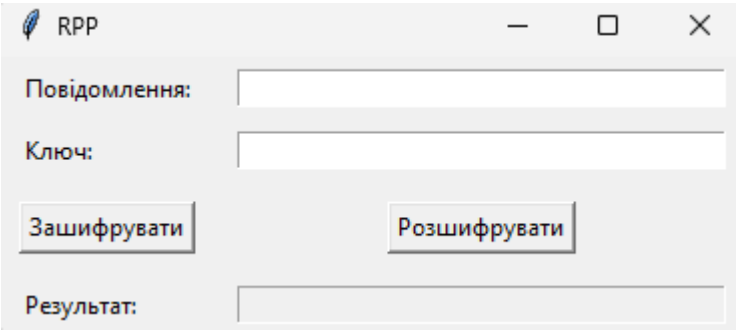

Рисунок 3.3 – головне меню середовища RPP

Тестування роботи середовища проводиться на підготовлених наборах тестів і представлені в таблицях 3.1, 3.2, 3.3

Таблиця 3.1 – Результати шифрування повідомлення

	Ключ	Повідомлення	Результат
Тест 1	54eew	HELLO	HJTTI
Тест 2	ghfne321	TERMOTIRE	NDOBQTDTC
Тест 3	45hfrjke34	IMAGE34GFJ	FCHZN34YHE

Таблиця 3.2 – Результати дешифрування повідомлення

	Ключ	Повідомлення	Результат
Тест 1	54eew	HJTTI	HELLO
Тест 2	ghfne321	NDOBQTDTC	TERMOTIRE
Тест 3	45hfrjke34	FCHZN34YHE	IMAGE34GFJ

Рис. 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 – Демонстрація тестової роботи програмного середовища RPP

На рис. 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 представлено знімок екрана основного вікна розробленого програмного середовища RPP під час тестової демонстрації його функціональності. Цей інтерфейс призначений для зручного виконання операцій шифрування та дешифрування текстових даних.

Головне вікно програми організовано таким чином, щоб користувач міг легко взаємодіяти з ключовими компонентами системи. У лівій частині розташоване текстове поле, позначене як "Повідомлення", куди користувач вводить дані, які підлягають шифруванню або дешифруванню Праворуч від нього знаходиться відповідне текстове поле "Результат", призначене для відображення результату обробки – зашифрованого тексту після операції шифрування або дешифрованого відкритого тексту.

Між цими полями розміщено дві функціональні кнопки: "Зашифрувати" та "Розшифрувати". Їх активація ініціює відповідні криптографічні перетворення. Нижче цих кнопок знаходиться поле введення "Ключ:", що дозволяє користувачеві ввести секретний ключ, необхідний для виконання як операції шифрування, так і дешифрування.

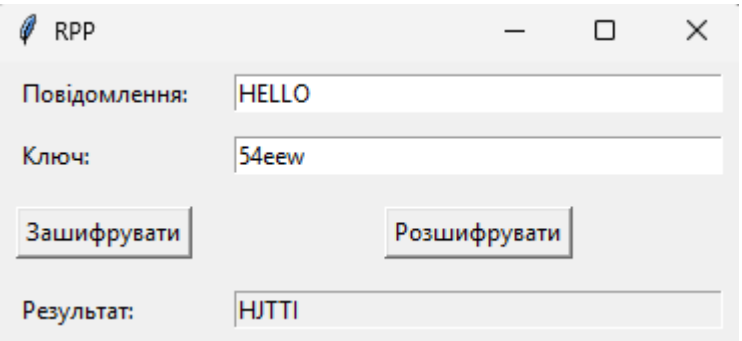

Рисунок 3.3 — Демонстрація роботи шифрування у першому тесті

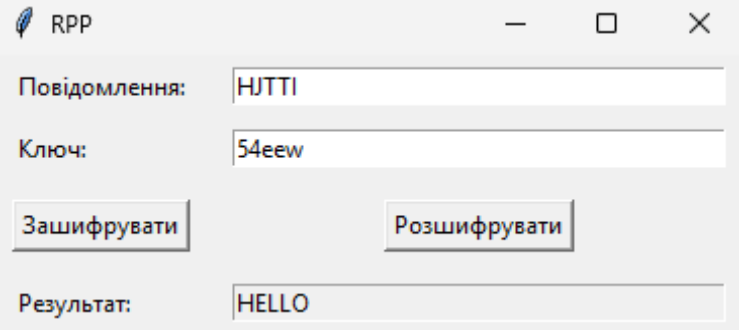

Рисунок 3.4 — Демонстрація роботи дешифрування у першому тесті

На рис 3.3, 3.4 демонструється приклад шифрування, де "HELLO"/"HJTTI" після натискання кнопки "Зашифрувати" перетворився на "HJTTI"/"HELLO".Це наочно ілюструє успішну роботу реалізованого алгоритму шифрування/дешифрування в межах розробленого програмного середовища RPP.

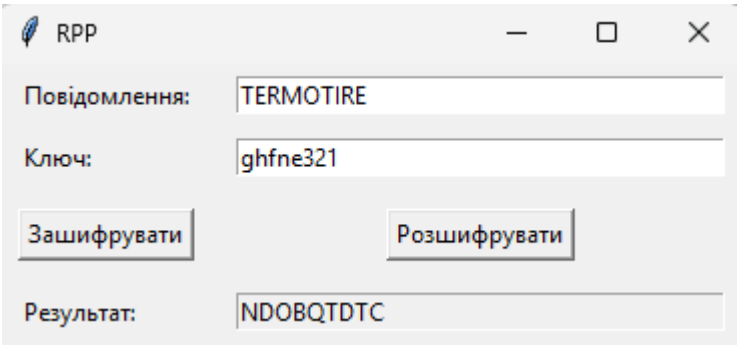

Рисунок 3.5 — Демонстрація роботи шифрування у другому тесті

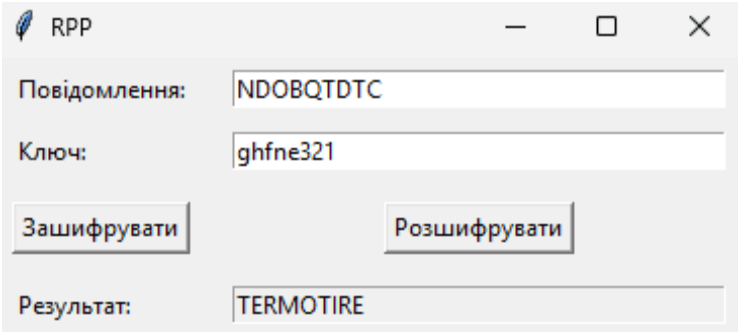

Рисунок 3.6 — Демонстрація роботи дешифрування у другому тесті

На рис 3.5, 3.6 демонструється приклад шифрування, де "TERMOTIRE"/"NDOBQTDTC" після натискання кнопки "Зашифрувати" перетворився на " NDOBQTDTC "/" TERMOTIRE ".Це наочно ілюструє успішну роботу реалізованого алгоритму шифрування/дешифрування в межах розробленого програмного середовища RPP.

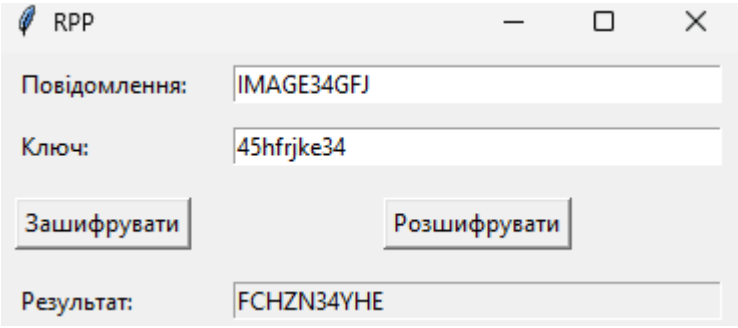

Рисунок 3.7 — Демонстрація роботи шифрування у третьому тесті

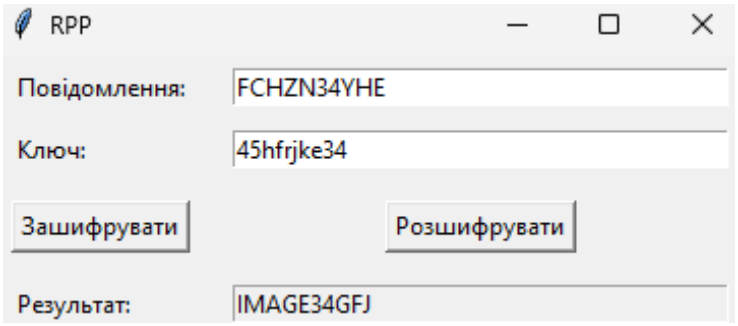

Рисунок 3.8 — Демонстрація роботи дешифрування у третьому тесті

На рис 3.7, 3.8 демонструється приклад шифрування, де "IMAGE34GFJ"/"FCHZN34YHE" після натискання кнопки "Зашифрувати" перетворився на " FCHZN34YHE "/" IMAGE34GFJ ".Це наочно ілюструє успішну роботу реалізованого алгоритму шифрування/дешифрування в межах розробленого програмного середовища RPP.

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи було детально розглянуто процес розробки програмного середовища RPP (Розробка Програмного Середовища), що є центральною практичною частиною дослідження. Глави цього розділу охоплюють вибір та характеристику криптографічного алгоритму, обґрунтування вибору інструментів розробки, опис архітектури програмного рішення та демонстрацію його тестової роботи.

Для реалізації функціоналу шифрування/дешифрування було обрано шифр Бофорта. Його вибір обґрунтований тим, що, будучи класичним поліалфавітним симетричним шифром, він дозволяє наочно продемонструвати принципи роботи симетричної криптографії та модульної арифметики. Попри його історичну природу та відомі вразливості до криптоаналізу при некоректному використанні, простота алгоритму Бофорта робить його ідеальним навчальним та демонстраційним прикладом для реалізації у програмному середовищі.

Описана архітектура програмного середовища RPP передбачає інтуїтивну взаємодію з користувачем через графічний інтерфейс. Була деталізована логіка функціонування програми, що включає введення тексту та ключа, активацію функцій шифрування/дешифрування та послідовну обробку символів повідомлення. Діаграма кінцевого автомату (рис. 3.2) наочно візуалізує переходи станів системи під час обробки даних, підкреслюючи її логічну структуру.

У розділі 3.4 представлена демонстрація тестової роботи програмного середовища RPP. За допомогою серії тестів (Табл. 3.1, 3.2 та Рис. 3.3-3.8) було підтверджено коректність виконання операцій шифрування та дешифрування для різних вхідних даних та ключів. Це наочно ілюструє повну функціональність розробленого програмного продукту та його готовність до використання згідно з поставленими задачами.

Таким чином, у даному розділі було успішно здійснено проєктування, реалізацію та тестування програмного середовища RPP, що повністю підтверджує практичну цінність та досягнення мети даної кваліфікаційної роботи у створенні функціонального інструменту для шифрування та дешифрування інформації.

ВИСНОВКИ

У кваліфікаційній роботі було всебічно досліджено та розроблено програмне середовище для шифрування та дешифрування інформації, що є актуальним у сучасному цифровому світі, де кіберзагрози постійно зростають.

Було розглянуто ключові принципи інформаційної безпеки, що забезпечуються криптографічними методами, а саме: конфіденційність, цілісність, автентифікація та контроль доступу. Проаналізовано основні типи шифрування – симетричне (наприклад, AES, DES, 3DES) та асиметричне (RSA), їхні переваги та недоліки. Особливу увагу приділено принципам роботи алгоритмів дешифрування та рекомендаціям щодо запобігання несанкціонованому доступу до конфіденційної інформації. Досліджено останні розробки в сфері криптографії, зокрема постквантову криптографію, гомоморфне шифрування та використання криптографії в технологіях блокчейн. Крім того, проведено порівняльний аналіз класичних шифрів Хілла, Віженера та Бофорта, що демонструють різні підходи до шифрування, хоча й мають переважно історичне значення у контексті сучасних вимог безпеки . Підкреслено вимоги до сучасних алгоритмів шифрування, такі як стійкість до криптоаналізу, ефективність, гнучкість та відсутність бекдорів.

Також було проведено огляд існуючих програм та середовищ для шифрування даних. Розглянуто програми для шифрування файлів і папок, такі як Encrypting File System (EFS) [3], VeraCrypt [5], BitLocker [6], FileVault та AxCrypt, а також можливості 7-Zip з шифруванням AES-256. Проаналізовано засоби шифрування повідомлень та електронної пошти, включаючи PGP/GPG, ProtonMail [9], Signal [8] та Telegram. Окрему увагу приділено програмам шифрування трафіку та VPN, зокрема OpenVPN та WireGuard [10], а також анонімній мережі Tor . Проведено огляд програм шифрування баз даних та хмарних сховищ, таких як Cryptomator [7] та власні рішення хмарних провайдерів. Завершенням другого розділу став аналіз недоліків та переваг різних середовищ шифрування даних.

Було виконано розробку програмного середовища RPP. Обрано та обґрунтовано характеристику шифру для реалізації в середовищі RPP. Детально описано інструменти розробки, які були використані для створення програмного продукту. Представлено діаграму кінцевого середовища RPP та продемонстровано його тестову роботу.

Таким чином, у ході виконання кваліфікаційної роботи досягнуто поставленої мети – розроблено програмне середовище для шифрування та дешифрування інформації. Проаналізовані теоретичні основи криптографії та існуючі програмні рішення дозволили створити ефективний інструмент, що відповідає сучасним вимогам до захисту даних. Розроблене середовище RPP демонструє практичну реалізацію криптографічних алгоритмів та може бути використане для підвищення рівня безпеки конфіденційної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Філосовські питання інформаційної безпеки: монографія / О.Г. Фільшін. – Суми: СумДУ, 2011. – 156 с. – URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/20822/1/Filsh.pdf (дата звернення: 16.06.2025)

2. Лабораторна робота 5. Симетричні системи шифрування // https://www.google.com/search?q=Kursoviks.com.ua – URL: https://ua.kursoviks.com.ua/metodychki/390-laboratorna-robota-5-simetrichni-sistemi-shifruvannya (дата звернення: 16.06.2025)

3. Encrypting File System – Вікіпедія. – URL: https://uk.wikipedia.org/wiki/Encrypting_File_System (дата звернення: 16.06.2025)

4. Захист інформації та конфіденційність даних: матеріали конф. – ПолтНТУ, 2015. – С. 207–208. – URL: https://reposit.nupp.edu.ua/bit (дата звернення: 16.06.2025)

5. VeraCrypt. Офіційний сайт. – URL: https://www.veracrypt.fr/ (дата звернення: 16.06.2025)

6. BitLocker Drive Encryption – Microsoft Docs. – URL: https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-overview (дата звернення: 16.06.2025)

7. Cryptomator. Secure cloud storage. – URL: https://cryptomator.org/ (дата звернення: 16.06.2025)

8. Signal Messenger. – URL: https://signal.org/ (дата звернення: 16.06.2025)

9. ProtonMail – Encrypted Email Service. – URL: https://protonmail.com/ (дата звернення: 16.06.2025)

10. WireGuard. – Офіційний сайт. – URL: https://www.wireguard.com/ (дата звернення: 16.06.2025)

11. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2),

120-126. – URL: https://people.csail.mit.edu/rivest/Rsapaper.pdf (дата звернення: 16.06.2025)

12. FIPS PUB 197. (2001). *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST). – URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (дата звернення: 16.06.2025)

13. Data Encryption Standard (DES). (1993). *FIPS PUB 46-3*. National Institute of Standards and Technology (NIST). – URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.46-3.pdf (дата звернення: 16.06.2025)

14. ISO/IEC 18033-1:2015. (2015). *Information technology — Security techniques — Encryption algorithms — Part 1: General*. International Organization for Standardization. – URL: https://www.iso.org/standard/66205.html (дата звернення: 16.06.2025)

15. Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons.

16. The Tor Project. – Офіційний сайт. – URL: https://www.torproject.org/ (дата звернення: 16.06.2025)

17. OpenVPN. – Офіційний сайт. – URL: https://openvpn.net/ (дата звернення: 16.06.2025)

18. Apple Support. (2024). *Use FileVault to encrypt your Mac startup disk*. – URL: https://support.apple.com/en-us/HT204837 (дата звернення: 16.06.2025)

19. AxCrypt. – Офіційний сайт. – URL: https://www.axcrypt.net/ (дата звернення: 16.06.2025)

20. 7-Zip. – Офіційний сайт. – URL: https://www.7-zip.org/ (дата звернення: 16.06.2025)

21. PGP. – Офіційний сайт. – URL: https://www.pgp.com/ (дата звернення: 16.06.2025)

22. GnuPG. – Офіційний сайт. – URL:https://gnupg.org/ (дата звернення: 16.06.2025)

23. Telegram. – Офіційний сайт. – URL: https://telegram.org/faq#q-what-is-secret-chats-and-how-do-they-work (дата звернення: 16.06.2025)

24. Boneh, D., & Shoup, V. (2017). *A Graduate Course in Applied Cryptography*.– URL:https://crypto.stanford.edu/~dabo/cryptobook/ (дата звернення: 16.06.2025)

25. Homomorphic Encryption Standard. (2024). *Homomorphic Encryption*. URL: https://homomorphicencryption.org/ (дата звернення: 16.06.2025)

26. National Institute of Standards and Technology. (2024). *Post-Quantum Cryptography*. – URL: https://csrc.nist.gov/projects/post-quantum-cryptography (дата звернення: 16.06.2025)