Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Божка Дмитра Валерійовича

на здобуття ступеня вищої освіти Бакалавра

Система шифрування файлів з різним ступенем таємності

Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітня програма Кібербезпека

Шифр КРБКБ.2101110.21.01.03 ПЗ

Виконав студент 4 курсу група КБ-21-1 03.06.25 Дмитро БОЖОК
Керівник канд. техн. наук, доцент Віра ТІТОВА
Нормоконтролер старший викладач Сергій МОСТОВИЙ

До захисту допускаю:
Завідувач кафедри кібербезпеки Юрій КЛЬОЦ

3 06 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Бакалавр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека

Освітня програма Кібербезпека

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

15 лютого 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Божку Дмитру Валерійовичу

1 Тема роботи Система шифрування файлів з різним ступенем таємності

Керівник роботи канд. техн. наук, доцент, Тітова Віра Юріївна

Затверджено наказом ректора університету від 7 лютого 2025 № 23

2 Строк подання студентом кваліфікаційної роботи на кафедру 3 червня 2025 р.

3 Вихідні дані до роботи: актуальні алгоритми шифрування, вимоги до зберігання секретних даних, класифікація інформації за рівнем таємності, приклади багаторівневого доступу

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз сучасних методів шифрування

Обґрунтування вибору алгоритмів

Архітектура та проєктування системи

Розробка

Тестування та оцінка безпеки системи

Висновки

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Презентація

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 16 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Вибір і затвердження теми кваліфікаційної роботи	лютий	
Ознайомлення з предметною областю	лютий	
Дослідження існуючих рішень	лютий	
Постановка задачі	березень	
Визначення загальних принципів рішення задачі	березень	
Деталізація принципів рішення задачі	квітень	
Розробка політик експлуатації і безпеки	квітень	
Оформлення пояснювальної записки згідно вимог	травень	
Оформлення графічної частини	травень	
Захист КР	червень	

Студент _____ Дмитро БОЖОК

Керівник кваліфікаційної роботи _____ Віра ТІТОВА

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Система шифрування файлів з різним ступенем таємності».

Автор роботи: Божок Дмитро Валерійович.

Керівник роботи: Тітова Віра Юріївна.

Пояснювальна записка: 72 с., 2 додатки, 24 рис., 3 таблиці, 40 джерел.

Графічна частина: 9 презентаційних слайдів.

ШИФРУВАННЯ ФАЙЛІВ, РІВЕНЬ ТАЄМНОСТІ, КРИПТОГРАФІЧНІ АЛГОРИТМИ, ІНФОРМАЦІЙНА БЕЗПЕКА, ГІБРИДНІ МЕТОДИ, КЛАСИФІКАЦІЯ ДАНИХ, СИСТЕМА ЗАХИСТУ.

У даній дипломній роботі розроблено концепцію та реалізовано систему шифрування файлів з урахуванням їхнього ступеня таємності. Проведено класифікацію інформаційних ресурсів за рівнями критичності та визначено відповідні криптографічні алгоритми, які найкраще підходять для кожного з них. Зокрема, розглянуто використання симетричних та асиметричних методів шифрування, а також гібридні підходи, що поєднують їх переваги.

У процесі роботи було реалізовано програмну модель, яка дозволяє здійснювати шифрування та дешифрування файлів з урахуванням політик безпеки, автентифікації користувачів та ведення журналу дій. Проведено тестування системи та проаналізовано її продуктивність, масштабованість і рівень захисту.

Результати дипломної роботи можуть бути використані в організаціях, що працюють з критичними даними, для побудови безпечних середовищ зберігання та передачі інформації.

01. 06 2025

ABSTRACT

Subject of qualification work: File encryption system with varying degrees of secrecy.

Author: Bozhok Dmytro Valeriiovych.

Head of work: Titova Vira Yuriivna.

Explanatory note: 72 p., 2 appendices, 24 figures, 3 tables, 40 sources.

Graphic part: 9 presentation slides.

FILE ENCRYPTION, LEVEL OF SECRECY, CRYPTOGRAPHIC ALGORITHMS, INFORMATION SECURITY, HYBRID METHODS, DATA CLASSIFICATION, SECURITY SYSTEM.

In this thesis, the concept and implementation of a file encryption system based on their degree of secrecy is developed. The information resources are classified by levels of criticality and the appropriate cryptographic algorithms are determined that are best suited for each of them. In particular, the use of symmetric and asymmetric encryption methods, as well as hybrid approaches that combine their advantages, are considered.

In the course of the work, a software model was implemented that allows encrypting and decrypting files, taking into account security policies, user authentication, and logging. The system was tested and its performance, scalability, and level of protection were analyzed.

The results of the thesis can be used in organizations working with critical data to build secure environments for storing and transmitting information

01.06.2025 

ЗМІСТ

ВСТУП

В умовах сучасного розвитку інформаційних технологій безпека даних набуває все більшої актуальності. Збільшення обсягів конфіденційної інформації, що передається через комп'ютерні мережі, зумовлює необхідність створення ефективних методів захисту даних. Одним із найважливіших способів забезпечення інформаційної безпеки є шифрування, яке дозволяє обмежити доступ до даних лише для авторизованих користувачів.

Шифрування файлів є одним із ключових аспектів кібербезпеки, оскільки воно дозволяє запобігти несанкціонованому доступу до критично важливої інформації. У сучасному світі дані можуть бути об'єктом атак з боку зловмисників, урядових організацій та конкурентів, що підвищує потребу в надійних алгоритмах шифрування. Різні рівні секретності інформації (наприклад, "таємно", "цілком таємно") визначають вимоги до методів її захисту, що є основою для класифікації систем шифрування.

У дипломній роботі розглядається система шифрування файлів із різними ступенями таємності, яка дозволяє забезпечити багаторівневий захист інформації відповідно до її чутливості. Буде проведено аналіз сучасних методів шифрування, їх переваг та недоліків, а також запропоновано підхід до класифікації рівнів секретності даних. Особлива увага приділяється питанням практичного застосування методів криптографії для захисту інформації у державному та корпоративному секторах.

Метою роботи є розробка системи шифрування файлів, що дозволяє забезпечити різні рівні безпеки інформації відповідно до її рівня конфіденційності. Для досягнення цієї мети в роботі вирішуються наступні завдання:

– аналіз сучасних методів та алгоритмів шифрування файлів;

– визначення вимог до системи шифрування з урахуванням різних ступенів таємності інформації;

– розробка архітектури системи шифрування файлів з можливістю налаштування рівнів безпеки;

– тестування запропонованого підходу на практичних прикладах та оцінка його ефективності;

Об’єктом дослідження є технології та методи шифрування файлів, що використовуються для захисту конфіденційної інформації. Предметом дослідження є принципи розробки системи шифрування файлів із можливістю адаптації під різні рівні секретності.

Практичне значення дослідження полягає у створенні ефективного підходу до класифікації та захисту інформації залежно від її рівня конфіденційності. Запропонована система може бути використана в державних установах, фінансових організаціях та компаніях, що працюють із конфіденційною інформацією.

Структура роботи включає три розділи. У першому розділі розглядаються основи теорії шифрування та сучасні методи захисту інформації. У другому розділі аналізуються особливості систем шифрування, що використовуються в державному та корпоративному секторах. У третьому розділі пропонується розробка системи шифрування файлів із різним ступенем секретності, проводиться тестування та аналіз її ефективності.

1 ТЕОРЕТИЧНІ ОСНОВИ ШИФРУВАННЯ ФАЙЛІВ

1.1 Аналіз предметної області і виявлення наявних проблем та завдань

У сучасному цифровому світі інформація є одним із найцінніших ресурсів. Захист даних від несанкціонованого доступу, модифікації або знищення є критично важливим завданням для урядових установ, комерційних компаній та приватних осіб. Одним з найефективніших методів захисту інформації є шифрування, яке забезпечує конфіденційність і цілісність даних. Проте, різні категорії інформації потребують різного рівня захисту залежно від ступеня її таємності.

Система шифрування файлів з різним ступенем таємності є складовою частиною інформаційної безпеки. Основна мета такої системи — гарантувати захищеність даних відповідно до їх рівня конфіденційності.

Класифікація даних може бути наступною:

– загальнодоступні дані;

– конфіденційні дані;

– секретні дані;

– цілком таємні дані.

Загальнодоступні дані – це інформація, яка не підлягає обмеженням у доступі та може бути відкрито використана будь-якими особами без необхідності отримання спеціального дозволу. Вони можуть бути опубліковані в інтернеті, на урядових порталах, у відкритих базах даних або поширюватися через ЗМІ та інші загальнодоступні джерела.

Прикладами загальнодоступних даних є:

– відкриті урядові дані;

– публічні реєстри;

– відкриті наукові дослідження та статті;

– дані про погодні умови, географічні карти, транспортні розклади;

– публічні профілі в соцмережах.

У більшості випадків загальнодоступні дані не потребують шифрування, оскільки вони призначені для відкритого використання. Проте існують випадки, коли навіть загальнодоступна інформація може підлягати захисту.

Захист від несанкціонованої модифікації — хоча загальнодоступні дані можуть бути відкритими, важливо гарантувати їхню достовірність. Наприклад, якщо хтось змінить офіційні статистичні показники або дані про вибори, це може призвести до серйозних наслідків. У таких випадках використовується цифровий підпис або контрольна сума (hash-функції) для перевірки цілісності даних.

Запобігання атакам "man-in-the-middle" — навіть якщо інформація є загальнодоступною, важливо захистити її під час передачі через інтернет, щоб уникнути її модифікації чи підміни зловмисниками. Наприклад, офіційні сайти використовують HTTPS для шифрування з'єднання між сервером і користувачем.

Агреговані загальнодоступні дані — окремі публічні дані можуть не містити загрози, але їхня комбінація може створити ризик. Наприклад, відкриті реєстри бізнесу у поєднанні з витоками особистих даних можуть бути використані для фішингових атак або шахрайства.

Якщо необхідно забезпечити цілісність або безпечну передачу загальнодоступної інформації, використовуються такі методи:

– цифрові підписи;

– контрольні суми;

– TLS/SSL (HTTPS);

– обмеження доступу до редагування.

Цифровий підпис — це електронні дані, отримані в результаті криптографічного перетворення інформації, які додаються до електронного документа або логічно з ним поєднуються. Він забезпечує підтвердження цілісності документа та ідентифікацію підписувача. Електронний цифровий підпис накладається за допомогою особистого (приватного) ключа та перевіряється за допомогою відкритого (публічного) ключа.

Для отримання електронного підпису необхідно звернутися до акредитованого центру сертифікації ключів (АЦСК). Процес включає подання

відповідних документів та генерацію особистого та відкритого ключів. Особистий ключ зберігається у підписувача на захищеному носії (наприклад, токені), а відкритий ключ публікується у сертифікаті, доступному для всіх бажаючих.

Використання цифрового підпису дозволяє підписувати електронні документи, забезпечуючи їх юридичну силу та автентичність. Це особливо актуально в умовах дистанційної роботи та електронного документообігу, де важливо забезпечити швидкий та безпечний обмін інформацією.

Контрольна сума — це значення, обчислене на основі набору даних за допомогою певного алгоритму, яке використовується для перевірки цілісності цих даних під час передачі або зберігання. Вона допомагає виявити помилки, що могли виникнути через збої в обладнанні, перешкоди в каналах зв'язку або інші фактори, що впливають на достовірність інформації.

Процес використання контрольної суми включає декілька етапів.

Обчислення контрольної суми на відправнику — перед передачею або збереженням даних відправник обчислює контрольну суму вихідного набору даних за допомогою обраного алгоритму.

Передача даних разом із контрольною сумою — обчислене значення контрольної суми додається до даних і разом з ними передається отримувачу або зберігається.

Перевірка на отримувачі — отримувач після прийому даних повторно обчислює контрольну суму отриманого набору даних і порівнює її з отриманою від відправника. Якщо значення збігаються, вважається, що дані не були пошкоджені; у протилежному випадку — виявлено помилку.

Існує декілька алгоритмів для обчислення контрольних сум, кожен з яких має свої особливості та сфери застосування.

Одним з прикладів є циклічний надлишковий код (CRC), який широко використовується в мережевих протоколах та зберіганні даних для виявлення випадкових помилок. Алгоритм обчислює залишок від ділення полінома, що представляє дані, на генераторний поліном.

Також можна відзначити алгоритм «Луна», його використовують для перевірки правильності ідентифікаційних номерів, таких як номери кредитних карт або міжнародні ідентифікатори мобільного обладнання (IMEI). Алгоритм допомагає виявити помилки при введенні номерів, забезпечуючи базову перевірку їх коректності.

Transport Layer Security та його попередник Secure Sockets Layer — це криптографічні протоколи, призначені для забезпечення безпечної передачі даних у мережі Інтернет. Вони забезпечують конфіденційність, цілісність та автентифікацію інформації, що передається між клієнтом та сервером. Коли ці протоколи використовуються разом із HTTP, утворюється HTTPS (HyperText Transfer Protocol Secure) — захищена версія протоколу HTTP, яка забезпечує шифрування даних під час їх передачі.

Протокол SSL був розроблений Американською компанією Netscape Communications у 1990-х роках для забезпечення безпечних з'єднань у веб-браузері Netscape Navigator. Після виявлення вразливостей у SSL було розроблено його наступника — TLS, який став стандартом для захищених комунікацій в Інтернеті. Остання версія, TLS 1.3, була опублікована в серпні 2018 року та включає покращення в безпеці та продуктивності.

Процес встановлення захищеного з'єднання за допомогою TLS/SSL складається з 3 основних етапів.

Рукостискання (Handshake) — клієнт і сервер обмінюються між собою повідомленнями для узгодження параметрів з'єднання, таких як версія протоколу, набір шифрів та обмін сертифікатами для автентифікації.

Генерація сесійного ключа — після автентифікації сторони генерують спільний сесійний ключ, який буде використовуватися для шифрування даних під час сеансу.

Шифрування даних — усі подальші дані, що передаються між клієнтом і сервером, шифруються за допомогою сесійного ключа, забезпечуючи конфіденційність та цілісність інформації.

Обмеження доступу до редагування — це процес встановлення контролю над тим, хто може вносити зміни до певних документів або файлів. Це забезпечує захист інформації від несанкціонованих змін і підтримує її цілісність.

Конфіденційна інформація — це дані, доступ до яких обмежено фізичною або юридичною особою, за винятком суб'єктів владних повноважень. Такі дані можуть поширюватися лише за згодою відповідної особи та на умовах, визначених нею.

В Україні конфіденційна інформація регулюється Законом "Про інформацію", де зазначено, що така інформація може поширюватися за бажанням відповідної особи у визначеному нею порядку. Також Закон "Про доступ до публічної інформації" визначає, що конфіденційна інформація — це відомості, доступ до яких обмежено фізичною або юридичною особою, крім суб'єктів владних повноважень.

Належний захист конфіденційної інформації є критично важливим для запобігання несанкціонованому доступу, розголошенню або використанню даних, що може призвести до фінансових втрат, репутаційних ризиків або юридичних наслідків для фізичних осіб та організацій. Впровадження ефективних заходів безпеки, таких як шифрування, контроль доступу та регулярний аудит інформаційних систем, допомагає забезпечити конфіденційність та цілісність даних.

Секретні дані — це інформація, розголошення якої може завдати шкоди національній безпеці держави. В Україні такі дані охоплюють відомості у сферах оборони, економіки, науки і техніки, зовнішніх відносин, державної безпеки та охорони правопорядку. Ця інформація спеціально охороняється державою і має обмежений доступ.

В Україні секретна інформація поділяється на три ступені секретності, які визначають рівень обмеження доступу та заходи захисту.

Особливої важливості — найвищий ступінь секретності, розголошення інформації цього рівня може призвести до надзвичайно серйозних наслідків для національної безпеки.

Цілком таємно — розголошення може завдати значної шкоди національним інтересам.

Таємно — найнижчий ступінь серед секретних даних, розголошення яких може завдати шкоди, але меншої, ніж у попередніх категоріях.

Процедура віднесення інформації до державної таємниці в Україні здійснюється через процедуру, яка включає аналіз потенційної шкоди національній безпеці у разі витоку інформації, визначення одного з трьох рівнів секретності залежно від важливості інформації та внесення до Зводу відомостей, що становлять державну таємницю (ЗВДТ).

Гриф секретності — це позначка на матеріальному носії інформації, яка вказує на ступінь її секретності. Надання відповідного грифа визначає рівень обмеження доступу та заходи безпеки, необхідні для захисту інформації.

Різні країни мають власні системи класифікації секретної інформації. Наприклад, у США існують три основні рівні секретності, «Confidential» (Таємно), «Secret» (Цілком таємно) та «Top Secret» (Особливої важливості). Кожен рівень визначає ступінь потенційної шкоди національній безпеці у разі розголошення інформації.

Розголошення або втрата документів, що містять державну таємницю, є серйозним правопорушенням і тягне за собою кримінальну відповідальність. Законодавство України передбачає покарання за такі дії, оскільки вони можуть завдати значної шкоди національній безпеці

Секретні дані відіграють ключову роль у забезпеченні національної безпеки. Їх належна класифікація, захист та контроль доступу є критично важливими для запобігання потенційним загрозам та забезпечення стабільності держави.

Шифрування даних є фундаментальним процесом у сфері інформаційної безпеки, що забезпечує конфіденційність, цілісність та захист інформації від несанкціонованого доступу.

Методи шифрування можуть включати в себе:

– симетричне шифрування;

– асиметричне шифрування;

– гібридні методи.

Симетричне шифрування — це метод криптографії, при якому для шифрування та дешифрування даних використовується один і той самий ключ. Цей підхід забезпечує високу швидкість обробки інформації та широко застосовується для захисту конфіденційних даних. До найпоширеніших алгоритмів симетричного шифрування належать AES, DES та RC4.

AES (Advanced Encryption Standard) — це симетричний блочний шифр, прийнятий урядом США як стандарт шифрування у 2001 році. Він оперує блоками даних розміром 128 біт і підтримує ключі довжиною 128, 192 або 256 біт. AES характеризується високою швидкістю та безпекою, що робить його популярним у різних додатках, включаючи захист даних у мережах та на пристроях зберігання інформації. Його структура базується на підстановках та перестановках, що забезпечує стійкість до різних криптоаналітичних атак.

DES (Data Encryption Standard) — це симетричний блочний шифр, розроблений у 1970-х роках. Він використовує блоки даних розміром 64 біти та ключ довжиною 56 біт.

Через відносно невелику довжину ключа DES вважається застарілим, оскільки сучасні обчислювальні потужності дозволяють здійснити повний перебір можливих ключів за прийнятний час. Для підвищення безпеки був розроблений алгоритм Triple DES (3DES), який застосовує DES тричі з різними ключами, збільшуючи ефективну довжину ключа та складність криптоаналізу.

RC4 — це симетричний потоковий шифр, розроблений Рональдом Рівестом у 1987 році. Він генерує псевдовипадковий потік байтів (кейстрім), який комбінується з відкритим текстом за допомогою операції XOR для отримання шифротексту.

RC4 був широко використовуваний у різних протоколах, таких як WEP та SSL/TLS. Однак з часом були виявлені вразливості в RC4, що робить його небажаним для використання в сучасних системах безпеки.

Порівняння всіх трьох методів шифрування наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння AES, DES та RC4

Алгоритм	Тип шифрування	Розмір блоку/потоку	Довжина ключа	Безпека
AES	Блочний	128 біт	128, 192, 256 біт	Висока
DES	Блочний	64 біти	56 біт	Низька (застарілий)
RC4	Потоковий	Потік байтів	Змінна (40-2048 біт)	Низька (вразливості виявлені)

Асиметричне шифрування, також відоме як криптографія з відкритим ключем, є методом шифрування, який використовує пару ключів: відкритий для шифрування та закритий для дешифрування. Цей підхід забезпечує високий рівень безпеки та широко використовується в сучасних комунікаційних системах.

На відміну від симетричного шифрування, де використовується один і той самий ключ для шифрування та дешифрування, асиметричне шифрування застосовує два різних, але математично пов'язаних ключі. Відкритий ключ доступний публічно і використовується для шифрування повідомлень, тоді як закритий ключ зберігається в таємниці власником і використовується для дешифрування. Це означає, що будь-хто може зашифрувати повідомлення для власника відкритого ключа, але лише власник закритого ключа може його розшифрувати.

RSA є одним з найвідоміших та найчастіше використовуваних алгоритмів асиметричного шифрування. Безпека RSA базується на складності факторизації великих чисел.

Незважаючи на свою популярність, RSA потребує використання великих ключів (наприклад, 2048 або 4096 біт) для забезпечення високого рівня безпеки, що може призводити до значних обчислювальних витрат.

Основні етапи роботи RSA (рисунок 1.1).

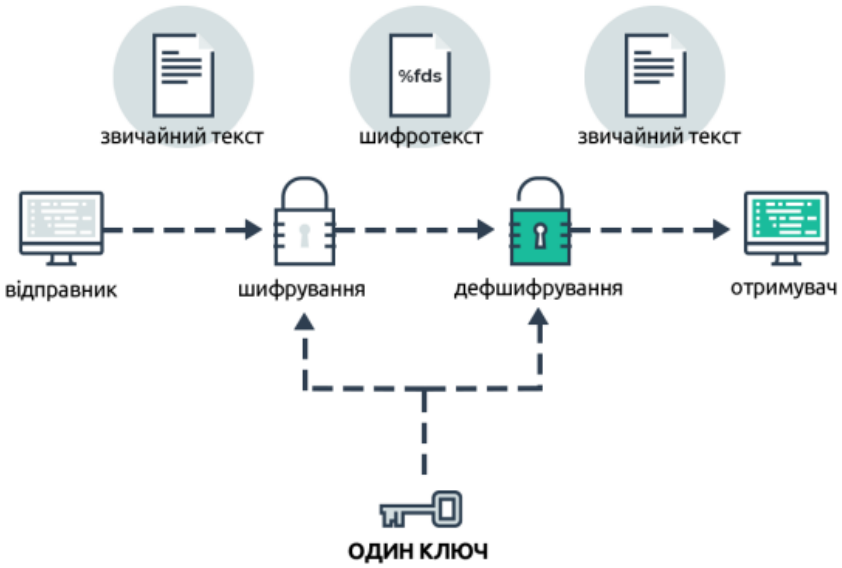


Рисунок 1.1 – Етапи роботи шифру RSA

ECC є сучасним підходом до асиметричного шифрування, який забезпечує аналогічний рівень безпеки з меншими розмірами ключів порівняно з RSA. Безпека ECC базується на складності задачі дискретного логарифмування на еліптичних кривих.

ECC має менші розміри ключів означають швидші обчислення та зменшене використання ресурсів, що особливо важливо для пристроїв з обмеженими можливостями, таких як мобільні телефони та смарт-карти. Покращена безпека, ECC з 256-бітним ключем забезпечує рівень безпеки, еквівалентний RSA з 3072-бітним ключем. Це означає, що для досягнення високого рівня безпеки ECC потребує значно менших ключів.

Обидва алгоритми вважаються безпечними при використанні відповідних розмірів ключів. Проте ECC пропонує більш високий рівень безпеки на біт ключа, що робить його стійкішим до деяких типів атак.

RSA має довшу історію використання та широко підтримується в різних системах та протоколах. Однак ECC швидко набирає популярності завдяки своїм перевагам.

Загрози безпеці:

– атаки «людина посередині» (Man-in-the-Middle, MITM);

– брутфорс-атаки для підбору ключів;

– використання шкідливого програмного забезпечення для викрадення ключів;

– недостатній контроль доступу до зашифрованих даних.

Попри значний розвиток криптографічних методів, у сфері шифрування файлів існують такі проблеми:

– необхідність гнучкої класифікації інформації, багато організацій стикаються з труднощами у визначенні рівня захисту для певних файлів;

– високі вимоги до продуктивності, складні алгоритми можуть значно уповільнювати роботу системи;

– керування ключами, проблема безпечного збереження, розповсюдження та ротації ключів шифрування;

– сумісність із різними системами, шифрування має підтримуватися на різних платформах (Windows, Linux, macOS, мобільні ОС);

– надмірна складність криптографічних систем може відлякувати користувачів і змушувати їх ігнорувати вимоги безпеки.

Виходячи з виявлених проблем, для розробки ефективної системи шифрування файлів із різним ступенем таємності необхідно вирішити такі завдання:

– розробити модель класифікації даних відповідно до рівня їх конфіденційності;

– обрати оптимальні алгоритми шифрування, що забезпечують баланс між продуктивністю та рівнем захисту;

– створити механізм безпечного управління ключами шифрування;

– реалізувати механізм контролю доступу до зашифрованих файлів;

– інтегрувати систему з популярними операційними системами та хмарними сховищами;

– забезпечити інтуїтивно зрозумілий інтерфейс користувача.

Таким чином, предметна область шифрування файлів є важливою частиною сучасної кібербезпеки. Незважаючи на значний розвиток методів криптографії, залишається низка викликів, що потребують вирішення. Визначення ключових проблем дозволяє сформулювати вимоги до розробки системи шифрування файлів із різним ступенем таємності, що підвищить безпеку даних та зручність їх використання.

1.2 Принципи побудови криптографічних систем

Криптографічні системи є основою інформаційної безпеки, забезпечуючи конфіденційність, цілісність та автентифікацію даних. Для ефективного функціонування вони повинні відповідати певним принципам побудови, які гарантують їхню надійність та захист від атак. Основні принципи криптографічних систем визначають методи та підходи до розробки та використання алгоритмів шифрування.

Один із фундаментальних принципів криптографії, сформульований Огюстом Керкгоффсом, стверджує, що безпека системи повинна ґрунтуватися не на секретності алгоритму, а на секретності ключа. Це означає, що навіть якщо зловмисник знає алгоритм шифрування, він не зможе отримати доступ до інформації без відповідного ключа.

Криптографічні системи повинні використовувати ключі достатньої довжини та складності, щоб унеможливити їхній підбір методом повного перебору (brute-force attack). Сучасні алгоритми шифрування передбачають використання ключів довжиною від 128 біт і більше, що робить їх стійкими до атак.

Якість криптографічного захисту залежить від використання випадкових або псевдовипадкових чисел у процесі генерації ключів. Генератори випадкових

чисел (True Random Number Generators, TRNG) забезпечують максимальну непередбачуваність ключів, що підвищує безпеку системи.

Ефективні криптографічні алгоритми повинні бути детермінованими, тобто однакові вхідні дані за однакових умов мають завжди давати однаковий зашифрований результат. Водночас вони повинні бути стійкими до змін у відкритому тексті (аваланч-ефект) – навіть незначна зміна у вхідних даних повинна суттєво змінювати вихідний результат.

Сучасні криптосистеми базуються на міжнародних стандартах шифрування, таких як AES (Advanced Encryption Standard), RSA, ECC (Elliptic Curve Cryptography) тощо. Це гарантує високу безпеку, оскільки такі алгоритми проходять багаторівневу перевірку та оптимізацію.

Окрім класичних атак, криптографічні системи повинні бути стійкими до атак на основі аналізу трафіку, коли зловмисник аналізує патерни передачі зашифрованих повідомлень. Для цього використовуються методи обфускації, шумові дані та інші технології приховування інформації.

Принципи побудови криптографічних систем відіграють ключову роль у забезпеченні їхньої надійності та безпеки. Дотримання цих принципів дозволяє створювати системи, що ефективно протистоять сучасним загрозам та атакам. У подальших розділах роботи будуть розглянуті конкретні реалізації криптографічних алгоритмів, їхні переваги та обмеження.

1.3 Види шифрування та їх застосування для захисту файлів

Шифрування файлів — це ключовий метод захисту інформації, який перетворює дані у форму, недоступну для несанкціонованого доступу. Воно дозволяє забезпечити конфіденційність та безпеку інформації в цифровому світі. Шифрування має широкий спектр застосувань, включаючи забезпечення безпеки інтернет-транзакцій, захист конфіденційних даних у сферах кібербезпеки та зберігання інформації, забезпечення безпечного обміну повідомленнями та захист

фінансових операцій. Існує кілька типів шифрування, кожен з яких має свої особливості та сфери застосування.

Першим типом шифрування є симетричне шифрування, при цьому типі шифрування використовується один і той самий ключ для шифрування та дешифрування даних. Цей метод є швидким та ефективним для обробки великих обсягів інформації.

Переваги симетричного шифрування:

– висока швидкість обробки даних;

– менше обчислювальних ресурсів порівняно з асиметричним шифруванням.

Недоліки симетричного шифрування:

– проблеми з безпечним обміном ключами;

– якщо ключ буде скомпрометовано, зловмисник отримає доступ до всіх зашифрованих даних.

Приклади алгоритмів, які використовують симетричний метод шифрування:

– AES (Advanced Encryption Standard);

– DES (Data Encryption Standard);

– 3DES (Triple DES);

– Twofish.

AES(advanced encryption system) також відомий як Rijndael, є одним із найпоширеніших алгоритмів шифрування. Був розроблений як альтернатива DES і після затвердження NIST у 2001 році став новим стандартом шифрування.

AES — це сімейство блокових шифрів з різною довжиною ключів та різними розмірами блоків.

AES працює методами підстановки та перестановки. Спочатку незашифровані дані перетворюються на блоки, а потім шифрування застосовується з використанням ключа. Процес шифрування складається з різних процесів, таких як зсув рядків, змішування стовпців і додавання ключів. Залежно від довжини ключа виконується 10, 12 чи 14 таких трансформацій (раундів). Варто

зазначити, що останній раунд відрізняється від попередніх і не включає підпроцес мікшування.

DES (Стандарт шифрування даних), представлений у 1976 році, є найстарішим методом симетричного шифрування. Він був розроблений IBM для захисту конфіденційних державних даних і був офіційно прийнятий у 1977 році для використання федеральними агентствами Сполучених Штатів. Алгоритм шифрування DES є одним із алгоритмів, включених у версії 1.0 і 1.1 TLS (безпека транспортного рівня).

DES перетворює 64-розрядні блоки відкритих текстових даних у зашифрований текст, розділяючи їх на два окремих 32-розрядних блоки, застосовуючи процес шифрування до кожного блоку окремо. Складається з 16 циклів різних процесів таких як розширення, перестановка, заміна чи інші операції за допомогою яких дані передаються в зашифрованому вигляді.

Симетричне шифрування широко використовується для захисту файлів, баз даних та інших великих обсягів інформації.

Другим видом шифрування є асиметричне шифрування — цей метод використовує публічний ключ для шифрування та приватний для дешифрування.

Переваги симетричного шифрування:

– безпечний обмін ключами;

– можливість використання цифрових підписів для перевірки автентичності даних.

Недоліки симетричного шифрування:

– нижча швидкість обробки даних порівняно з симетричним шифруванням.

Приклади алгоритмів, які використовують асиметричний метод шифрування:

– RSA (Rivest-Shamir-Adleman);

– ECC (Elliptic Curve Cryptography).

Асиметричне шифрування часто використовується для захисту електронної пошти, цифрових підписів та обміну ключами в безпечних протоколах.

Ще одним методом шифрування є гібридне шифрування, який поєднує переваги симетричного та асиметричного шифрування. Кожен алгоритм шифрування має свої недоліки. Наприклад, симетричне шифрування ідеально підходить для швидкого шифрування великих обсягів даних. Але він не пропонує підтвердження особи, що важливо, коли мова йде про безпеку в Інтернеті. З іншого боку, асиметричне шифрування дозволяє отримати доступ до даних одержувача. Однак ця перевірка значно сповільнює процес кодування. У цьому методі симетричне шифрування використовується для шифрування основних даних, а асиметричне для захисту симетричного ключа.

Переваги гібридного шифрування:

– висока швидкість обробки даних;

– безпечний обмін ключами.

Гібридне шифрування широко застосовується в протоколах безпечного зв'язку, таких як SSL/TLS.

Шифрування файлів є критично важливим для забезпечення конфіденційності та цілісності даних. Приклади переваг застосування шифрування:

– забезпечує захист особистих даних, фінансової інформації та інших конфіденційних відомостей від несанкціонованого доступу;

– використовується для захисту даних, що зберігаються на жорстких дисках, флеш-накопичувачах та в хмарних сервісах;

– під час передачі файлів через мережу шифрування забезпечує їх захист від перехоплення та несанкціонованого доступу;

– захист комерційної таємниці, внутрішньої документації та інших важливих даних.

Шифрування є невід'ємною частиною сучасної інформаційної безпеки, забезпечуючи захист даних як під час зберігання, так і при передачі. Вибір відповідного типу шифрування та інструментів є важливим та залежить від конкретних потреб та вимог до безпеки.

2 ШИФРУВАННЯ ФАЙЛІВ ВІДПОВІДНО ДО РІВНЯ ТАЄМНОСТІ

2.1 Порівняльний аналіз сучасних криптографічних алгоритмів

У сучасному цифровому світі, де обмін інформацією відбувається миттєво та масово, забезпечення конфіденційності, цілісності та автентичності даних стало критично важливим. Криптографія, як наука про методи захисту інформації, відіграє ключову роль у забезпеченні безпеки в інформаційних системах. Сучасні криптографічні алгоритми базуються на складних математичних концепціях, таких як теорія чисел, теорія ймовірностей та обчислювальна складність, що дозволяє створювати надійні механізми захисту даних.

Симетричне шифрування — використовує один і той самий ключ для шифрування та дешифрування даних. Цей підхід забезпечує високу швидкість обробки інформації, що робить його ефективним для захисту великих обсягів даних. Основними алгоритмами даного типу шифрування є:

– AES (Advanced Encryption Standard);

– DES (Data Encryption Standard);

– 3DES (Triple DES);

– Twofish.

AES (Advanced Encryption Standard) — це симетричний алгоритм блочного шифрування, який на сьогодні є одним з найпоширеніших і найнадійніших стандартів шифрування у світі. Його було обрано Національним інститутом стандартів і технологій США (NIST) у 2001 році як заміну застарілому стандарту DES (Data Encryption Standard).

Шифрування та дешифрування даних використовує один і той самий секретний ключ. Обидві сторони, що обмінюються зашифрованою інформацією, повинні мати однаковий секретний ключ. AES працює з фіксованими розмірами блоків даних. Стандартний розмір блоку для AES становить 128 біт (16 байт). AES підтримує три різні розміри ключів, 128біт, 192біт та 256 біт, чим більший розмір ключа, тим вищий рівень безпеки, оскільки збільшується кількість можливих ключів, що ускладнює їх підбір методом перебору. Процес шифрування в AES

складається з кількох повторюваних кроків, які називаються раундами. Кількість раундів залежить від розміру ключа.

Процес шифрування AES(рисунок 2.1):

– початковий раунд, в якому вхідний блок даних і початковий ключ комбінуються за допомогою операції XOR;

– основні раунди, де кожен раунд включає чотири основні етапи, такі як SubBytes, ShiftRows, MixColumns, AddRoundKey;

– останній раунд, який майже ідентичний основним раундам, за винятком відсутності кроку MixColumns.

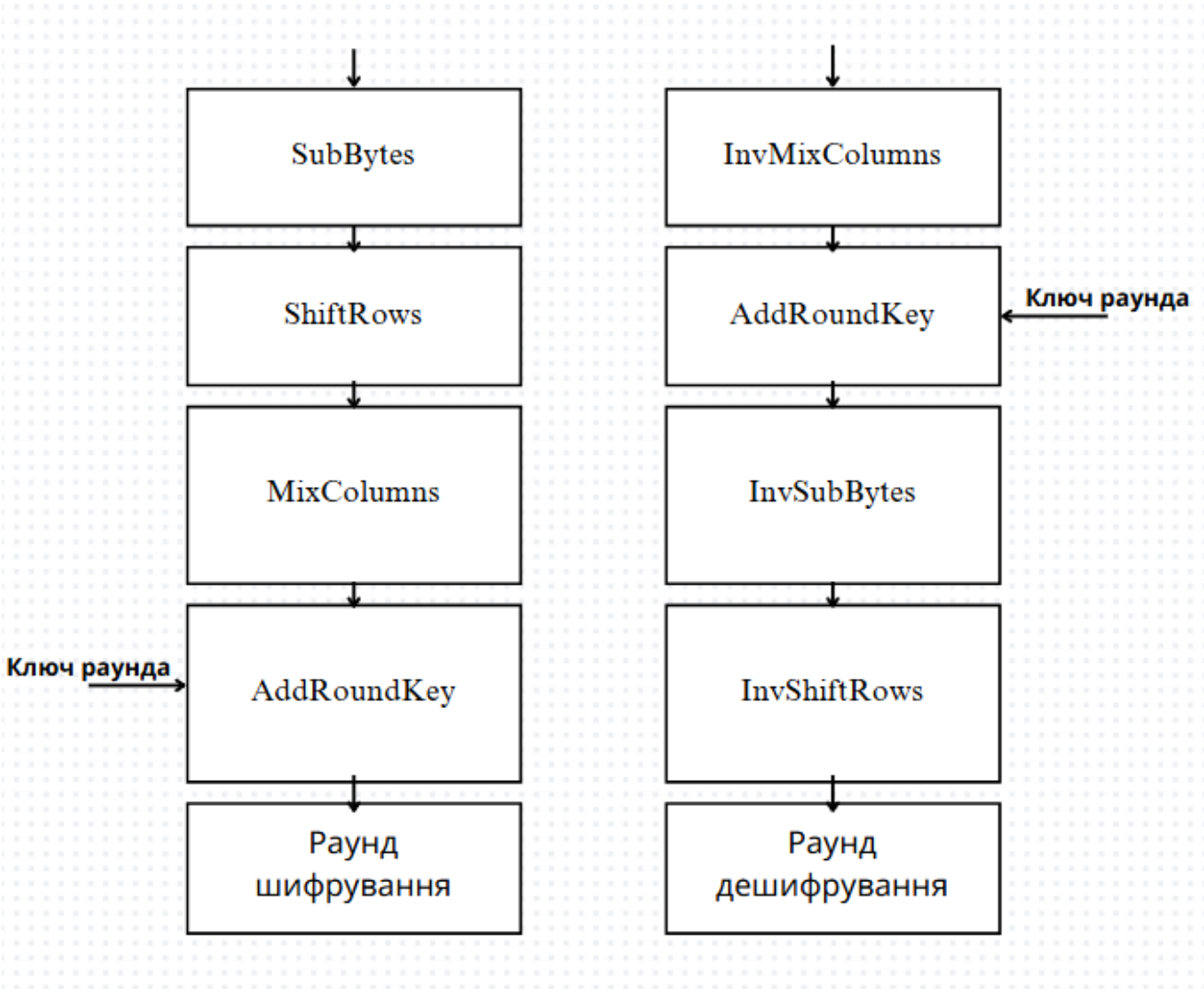


Рисунок 2.1 – Операції в раундах шифрування/дешифрування AES-128

SubBytes — кожен байт у блоці даних замінюється іншим байтом відповідно до фіксованої таблиці підстановки. Цей крок забезпечує нелінійність шифрування, що ускладнює лінійний криптоаналіз (рисунок 2.2).

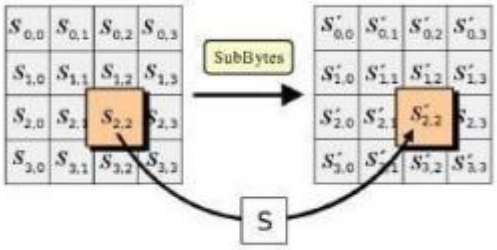


Рисунок 2.2 – Операція SubBytes

ShiftRows — зсув рядків, де байти в кожному рядку матриці стану циклічно зсуваються на певну кількість позицій вліво. Цей крок забезпечує дифузію, тобто вплив одного біта вхідного блоку на кілька бітів вихідного блоку (рисунок 2.3).

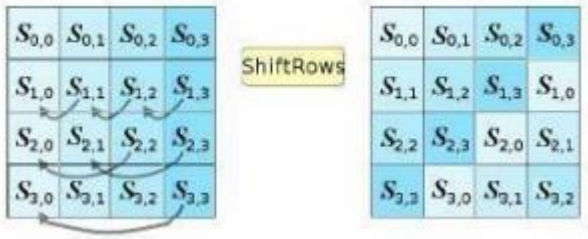


Рисунок 2.3 – Операція ShiftRows

MixColumns — кожен стовпець матриці стану перетворюється шляхом множення на фіксовану матрицю. Ця операція також сприяє дифузії, перемішуючи байти між стовпцями (рисунок 2.4).

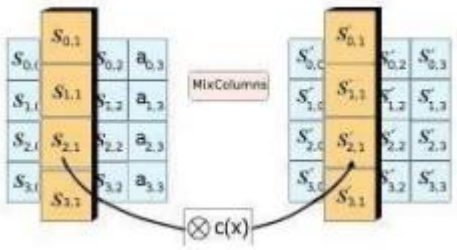


Рисунок 2.4 – Операція MixColumns

AddRoundKey — поточний стан блоку даних комбінується з раундовим ключем за допомогою операції XOR (рисунок 2.5).

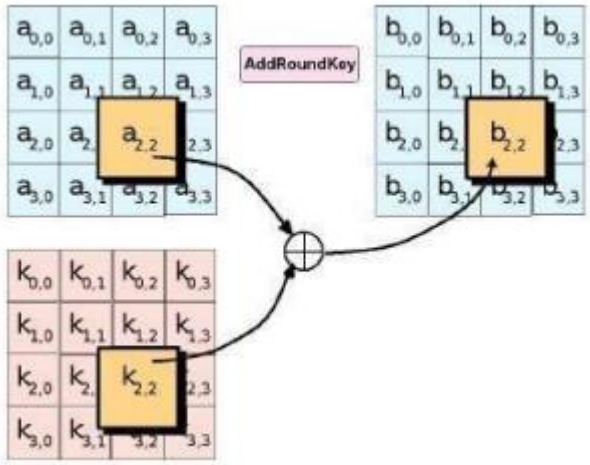


Рисунок 2.5 – Операція AddRoundKey

AES широко використовується в різноманітних застосуваннях, включаючи:

– шифрування файлів і дисків;

– захист інтернет-трафіку, VPN-з'єднань;

– захист Wi-Fi з використанням протоколу WPA2/WPA3;

– шифрування конфіденційних даних у базах даних;

– захист даних на смартфонах і планшетах;

– може використовуватися в деяких допоміжних процесах.

Безпека AES вважається дуже високою. На сьогодні не існує відомих практичних атак, які могли б зламати AES при використанні достатньо довгих ключів (128 біт і більше). Теоретичні атаки існують, але вони є складними та непрактичними для реалізації.

Вибір розміру ключа залежить від вимог до безпеки. AES-128 вважається достатньо безпечним для більшості застосувань, але для більш критичних систем може використовуватися AES-192 або AES-256 для забезпечення додаткового рівня захисту.

Підсумовуючи, AES є потужним, гнучким і широко прийнятим стандартом симетричного шифрування, який забезпечує високий рівень безпеки для

різноманітних цифрових даних і комунікацій. Його надійність і ефективність зробили його незамінним інструментом у сучасній криптографії.

DES (Data Encryption Standard) — був розроблений компанією IBM на початку 1970-х років на основі ранішого алгоритму Lucifer. Хоча сьогодні він вважається застарілим і ненадійним для багатьох сучасних застосувань, він зіграв важливу роль в історії криптографії.

Структура алгоритму DES показана на рисунку 2.6.

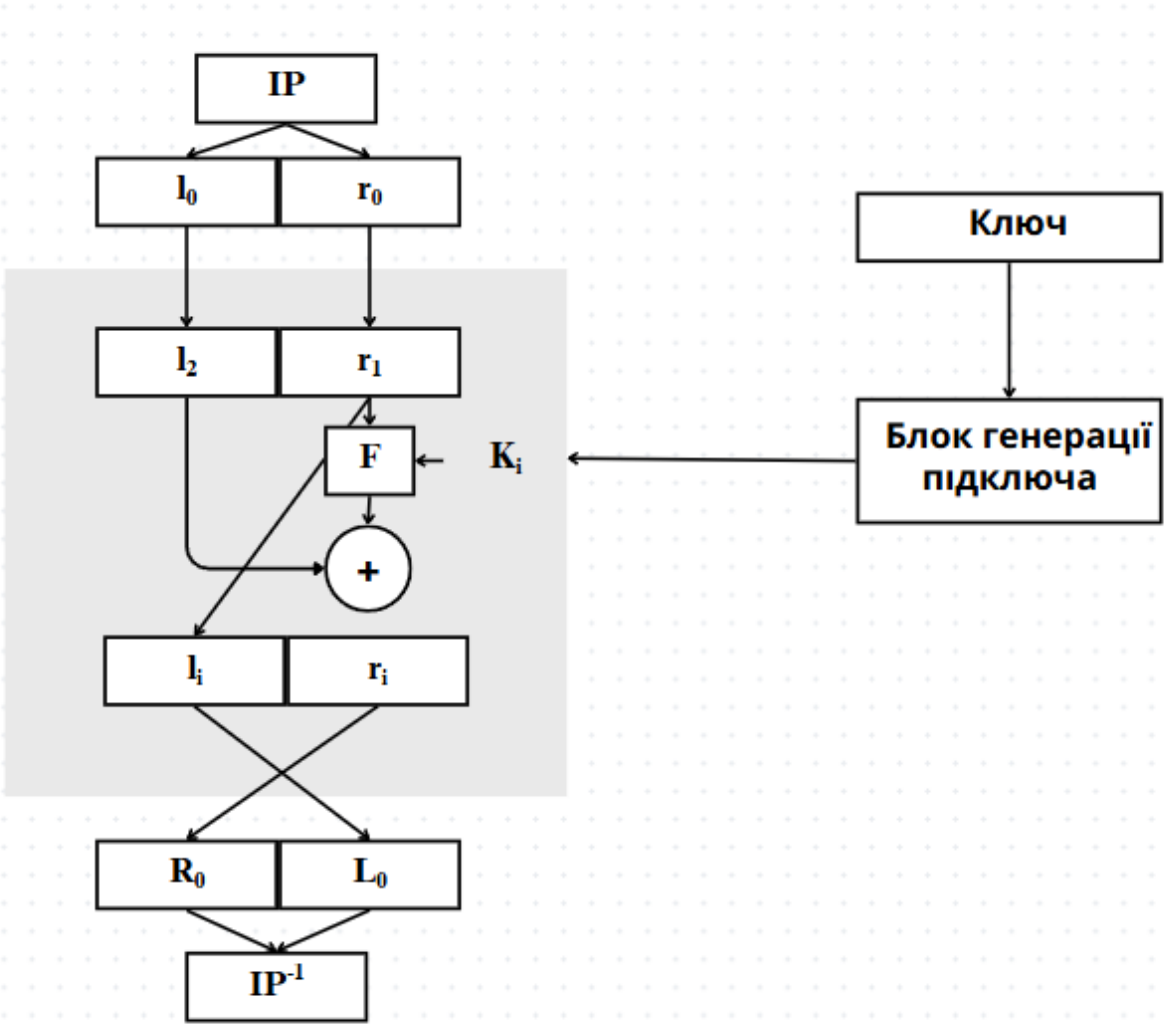


Рисунок 2.6 – Структура алгоритму DES

Основні характеристики DES:

– використовує один і той самий секретний ключ для шифрування та дешифрування даних;

– працює з блоками даних фіксованого розміру в 64 біти;

– довжина ключа DES становить 56 біт, хоча ключ часто представляється як 64-бітне число, 8 бітів використовуються для перевірки парності та ігноруються;

– структура Фейстеля;

– кінцева та початкова перестановки (рисунок 2.7-2.8).

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	36	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Рисунок 2.7 – Матриця початкової перестановки вхідного блоку

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Рисунок 2.8 – Матриця кінцевої перестановки вхідного блоку

Структура Фейстеля — у кожному раунді блок даних розділяється на дві половини (ліву та праву), і над правою половиною виконується складна функція (F-функція), яка включає підстановки (S-блоки) та перестановки. Результат цієї функції потім комбінується з лівою половиною за допомогою операції XOR, а потім ліва та права половини міняються місцями для наступного раунду.

Початкова та кінцева перестановки — першим раундом виконується початкова перестановка (IP) бітів вхідного блоку, а після 16-го раунду — обернена початкова перестановка (IP^{-1}) для отримання зашифрованого тексту. Схема роботи мережі Фейстеля показана на рисунку 2.9.

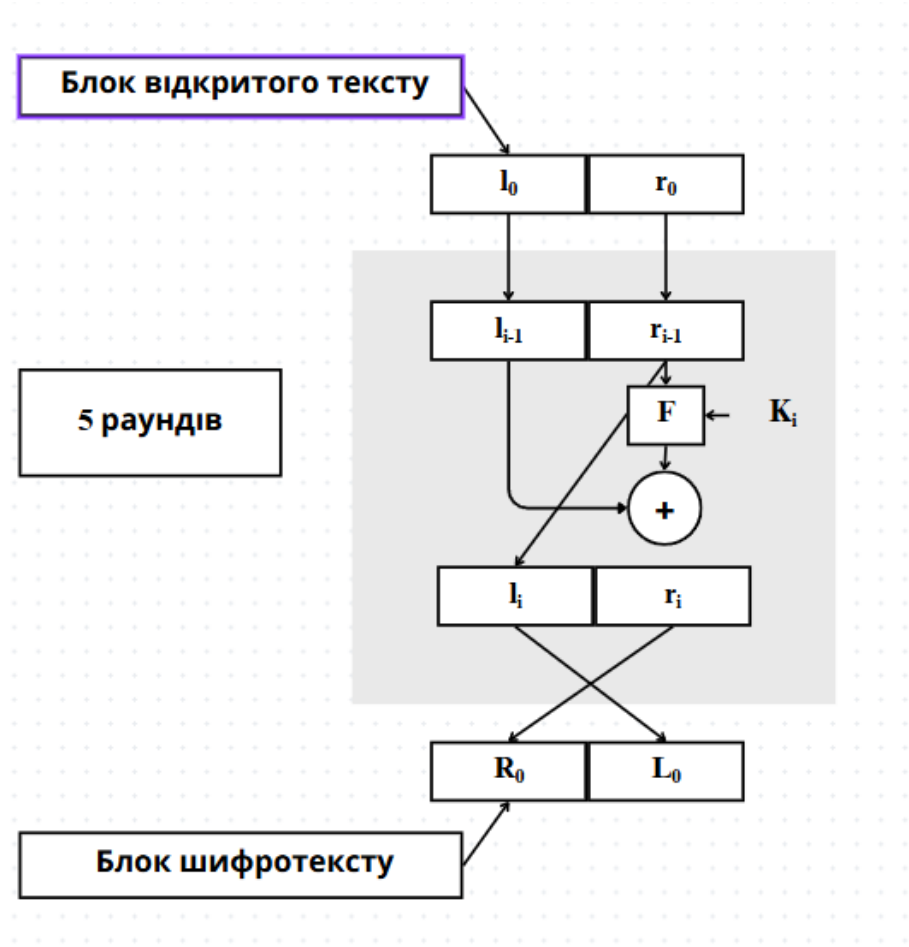


Рисунок 2.9 – Схема мережі Фейстеля

Процес шифрування DES:

– початкова перестановка;

– 16 раундів;

– кінцева перестановка.

Кожен із 16 раундів включає в себе такі кроки:

– розширення правої половини (32 біти) до 48 бітів за допомогою функції розширення (E);

– XOR розширеної правої половини з раундовим ключем (48 бітів), отриманим з головного ключа за допомогою розкладу ключів;

– проходження результату через вісім S-блоків, кожен з яких приймає 6 бітів на вхід і видає 4 біти на вихід, забезпечуючи нелінійність;

– Перестановка 32-бітного виходу S-блоків;

– XOR результату перестановки з лівою половиною;

– Обмін лівої та правої половин (крім останнього раунду).

З часом стало очевидно, що 56-бітний розмір ключа є недостатнім для забезпечення безпеки від атак методом перебору. З розвитком обчислювальної потужності стало можливим зламати DES за відносно короткий час. Одним з відомих прикладів є злом DES за 56 годин у 1998 році Electronic Frontier Foundation (EFF).

Крім того, були виявлені деякі інші потенційні слабкості DES, хоча жодна з них не є настільки серйозною, як мала довжина ключа.

DES був важливим кроком у розвитку сучасної криптографії та довгий час служив стандартом шифрування. Однак, через свій малий розмір ключа, він більше не вважається безпечним для більшості сучасних застосувань і був замінений на більш надійні алгоритми, такі як AES.

3DES (Triple DES) — розроблений як тимчасове рішення для підвищення безпеки оригінального стандарту DES після виявлення в ньому вразливостей, пов'язаних з малою довжиною ключа. 3DES застосовує алгоритм тричі з різними ключами, підвищуючи безпеку, але знижуючи продуктивність.

Основні характеристики:

– симетричний алгоритм, що використовує один і той самий ключ для шифрування та дешифрування, і працює з блоками даних фіксованого розміру в 64 біти;

– застосовує алгоритм DES тричі послідовно до кожного блоку даних;

– може використовувати два або три 56-бітні ключі DES.

Процес шифрування:

– вхідний 64-бітний блок даних шифрується за допомогою першого ключа алгоритмом DES;

– результат першого кроку дешифрується за допомогою другого ключа алгоритмом DES;

– результат другого кроку знову шифрується за допомогою третього ключа алгоритмом DES.

3DES значно безпечніший за оригінальний DES завдяки більшій ефективній довжині ключа, особливо при використанні трьох різних ключів, однак оскільки 3DES застосовує алгоритм DES тричі, він утричі повільніший за DES і значно повільніший за сучасні алгоритми, такі як AES. 3DES успадкував невеликий розмір блоку DES (64 біти), що може бути менш ефективним і потенційно вразливим до певних типів атак при шифруванні великих обсягів даних. Незважаючи на свої недоліки, 3DES довгий час залишався важливим стандартом шифрування і використовувався в багатьох протоколах та системах, включаючи:

– фінансові транзакції;

– VPN;

– урядові та військові застосування.

3DES був важливим еволюційним кроком у криптографії, забезпечивши підвищену безпеку порівняно з DES. З появою більш ефективних і безпечних алгоритмів, таких як AES, використання 3DES поступово зменшується. AES пропонує більші розміри ключів і блоків, а також кращу продуктивність. Хоча 3DES все ще може зустрічатися в деяких застарілих системах, для нових розробок рекомендується використовувати сучасніші алгоритми шифрування.

Twofish — це симетричний блочний шифр, який був одним з п'яти фіналістів конкурсу AES (Advanced Encryption Standard), але зрештою не був обраний стандартом. Однак, це потужний і добре проаналізований алгоритм, який має свої особливості та переваги.

Основні характеристики:

– використовує один і той самий секретний ключ для шифрування та дешифрування;

– має фіксований розмір блоку 128 біт;

– підтримує змінні розміри ключів 128, 192 та 256 біт;

– використовує модифіковану мережу Фейстеля з 16 раундами, яка включає додаткові елементи, такі як залежні від ключа S-блоки;

– використання S-блоків, які генеруються на основі секретного ключа, що робить криптоаналіз складнішим, оскільки структура S-блоків не є фіксованою та залежить від використовуваного ключа;

– має досить складний процес розкладу ключів, який генерує раундові ключі та використовується для створення залежних від ключа S-блоків.

Процес шифрування:

– попереднє відбілювання, коли вхідний 128-бітний блок даних комбінується з частинами ключа за допомогою операції XOR;

– 16 раундів, кожен раунд включає такі основні кроки як F-функція, перестановка PHT, додавання раундового ключа та обмін половин;

– післяраундове відбілювання, де після 16 раундів результат комбінується з іншими частинами ключа за допомогою операції XOR.

Візуальна схема шифрування та дешифрування відображена на рисунку 2.10.

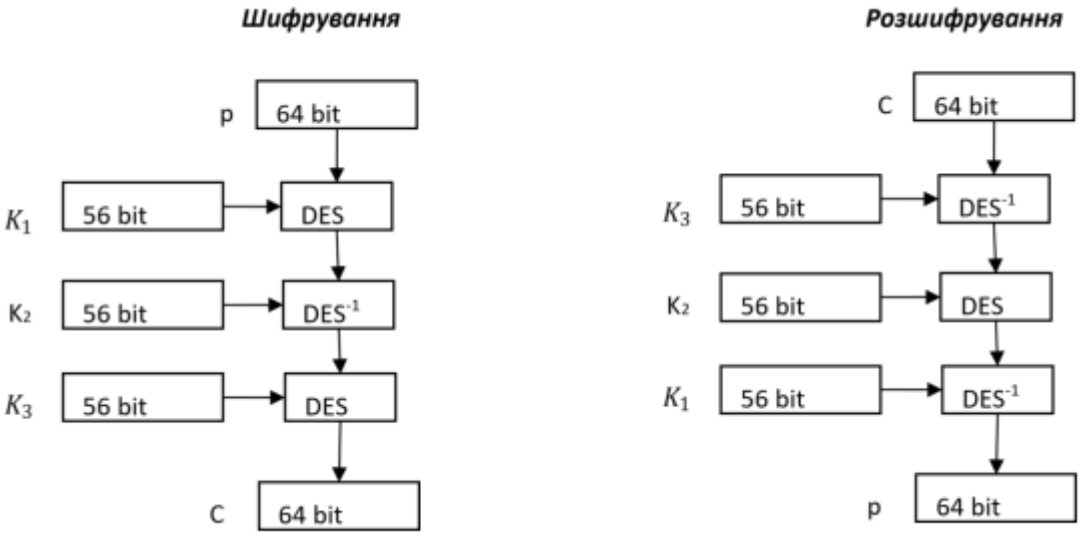


Рисунок 2.10 – Схема шифрування/дешифрування 3DES

Twofish вважається дуже безпечним алгоритмом. Його розробники провели значний аналіз безпеки алгоритму, і він також був ретельно вивчений криптографічною спільнотою. На сьогодні невідомо про практичні атаки, які б

могли ефективно зламати Twofish при використанні достатньо довгих ключів. Використання залежних від ключа S-блоків вважається однією з сильних сторін Twofish з точки зору безпеки.

Продуктивність Twofish є досить високою, хоча в програмній реалізації він може бути дещо повільнішим за AES на деяких платформах. Однак він добре підходить для апаратної реалізації.

Хоча Twofish не став стандартом AES, він використовується в деяких криптографічних бібліотеках і програмних продуктах. Він також розглядається як надійний альтернативний алгоритм шифрування. Приклади використання:

– програмне забезпечення для шифрування файлів і дисків;

– криптографічні бібліотеки;

– дослідження та розробка.

Асиметричне шифрування — використовує пару ключів, публічний для шифрування та приватний для дешифрування. Цей метод забезпечує високий рівень безпеки при передачі даних, особливо в відкритих мережах.

Основні алгоритми:

– RSA;

– ECC;

– DSA.

RSA (Rivest–Shamir–Adleman) є одним з перших і найпоширеніших алгоритмів асиметричного шифрування.

Основні характеристики:

– використовує пару ключів, де відкритий ключ використовується для шифрування а закритий ключ використовується для дешифрування;

– працює з блоками даних, розмір яких зазвичай залежить від довжини ключа;

– розміри ключів.

Безпека RSA ґрунтується на математичній складності факторизації великого складеного числа на два великих прості числа.

Принцип роботи RSA:

– генерація ключів;

– шифрування;

– дешифрування.

Приклади використання:

– протоколи TLS/SSL використовують RSA для встановлення безпечного з'єднання та обміну симетричними ключами;

– створення цифрових підписів, що дозволяє підтвердити автентичність та цілісність цифрових документів;

– шифрування електронної пошти;

– керування доступом.

Приклад шифрування за алгоритмом RSA схематично показано на рисунку 2.11.

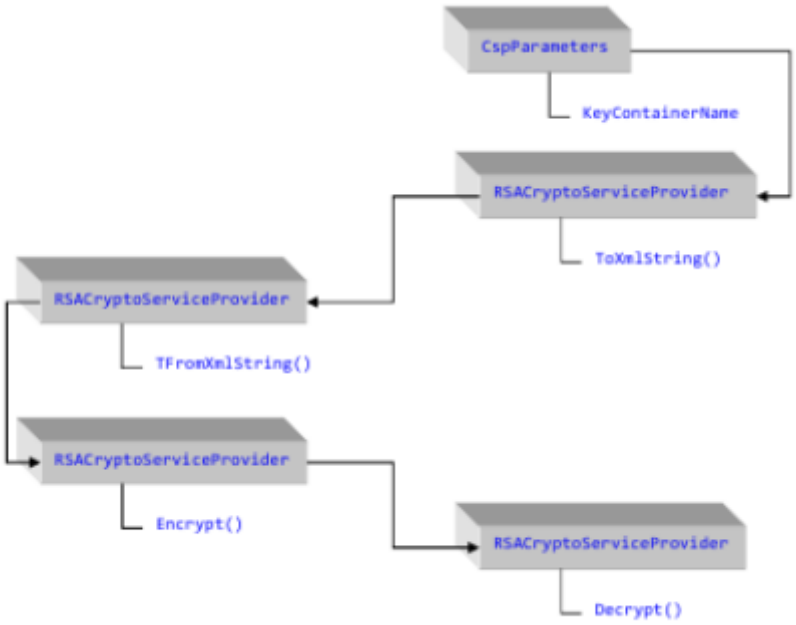


Рисунок 2.11 – Приклад реалізації алгоритму RSA

Переваги RSA:

– простота розуміння та реалізації;

– надійність;

– підтримка цифрових підписів;

– широке поширення.

Недоліки RSA:

– RSA є більш повільним алгоритмом порівняно з симетричними алгоритмами, такими як AES, особливо при шифруванні великих обсягів даних;

– вразливість до атак на основі факторизації;

– вразливість до певних типів атак

З огляду на загрозу з боку квантових комп'ютерів, які теоретично можуть ефективно факторизувати великі числа за допомогою алгоритму Шора, тривають дослідження в галузі постквантової криптографії. Розробляються нові асиметричні алгоритми, які, як очікується, будуть стійкими до атак квантових комп'ютерів і можуть у майбутньому замінити RSA для певних застосувань. Незважаючи на це, RSA залишається важливим і широко використовуваним криптографічним алгоритмом на сьогоднішній день, особливо для завдань, де потрібен обмін ключами та цифрові підписи.

Elliptic Curve Cryptography (ECC) — це сучасний підхід до асиметричної криптографії, який набув значної популярності завдяки своїй здатності забезпечувати високий рівень безпеки з використанням коротших ключів порівняно з традиційними алгоритмами, такими як RSA.

Основні характеристики:

– використовує відкритий ключ для шифрування та закритий ключ для дешифрування;

– безпека ECC ґрунтується на математичній складності задачі дискретного логарифмування на еліптичних кривих;

– використовує властивості еліптичних кривих над скінченними полями;

– можливість використання значно коротших ключів для досягнення того ж рівня безпеки, що й RSA.

Принцип роботи:

– вибір еліптичної кривої та базової точки;

– генерація ключів;

– шифрування;

– цифровий підпис.

Приклади застосування:

– протоколи безпеки Інтернету;

– мобільні пристрої;

– криптовалюти;

– бездротові мережі;

– інтернет речі.

Переваги:

– забезпечує еквівалентний рівень безпеки з набагато коротшими ключами, ніж RSA;

– швидші обчислення та менші вимоги до ресурсів;

– коротші ключі та підписи зменшують обсяг переданих даних.

Недоліки:

– математична основа є більш складною для розуміння порівняно з RSA;

– раніше існували деякі патентні обмеження, пов'язані з ECC, але більшість з них вже закінчилися;

– менша поширеність.

ECC є потужним і ефективним алгоритмом асиметричної криптографії, який пропонує високий рівень безпеки з меншими розмірами ключів. Його переваги роблять його особливо привабливим для сучасних застосувань, особливо в середовищах з обмеженими ресурсами.

DSA (Digital Signature Algorithm) — це стандарт Федерального уряду США, який застосовується для створення цифрових підписів, забезпечуючи автентичність та цілісність даних.

Основні характеристики:

– приватний ключ для створення підпису та публічний для його перевірки;

– призначений виключно для цифрових підписів;

– безпека базується на складності задачі дискретного логарифмування в скінченному полі;

– використовує набір публічних параметрів, які можуть бути спільними для групи користувачів.

Процес генерації ключів:

– обираються публічні параметри p, q і g;

– обирається випадкове ціле число x в діапазоні $0 < x < q$, яке є приватним ключем;

– обчислюється публічний ключ.

Процес створення цифрового підпису:

– генерується випадкове секретне число k в діапазоні $0 < k < q$;

– обчислюється значення r;

– обчислюється значення s;

– цифровий підпис для повідомлення є пара (r, s).

Безпека DSA залежить від складності задачі дискретного логарифмування в скінченному полі Z_p^*. Для забезпечення високого рівня безпеки необхідно використовувати достатньо великі значення p і q. Важливою умовою безпеки DSA є використання унікального випадкового числа k для кожного підпису. Якщо зловмиснику вдасться дізнатися значення k для одного підпису, він зможе обчислити приватний ключ x. Повторне використання одного і того ж значення k також може призвести до викриття приватного ключа.

Приклади застосування:

– протокол Digital Signature Standard (DSS);

– урядові та військові системи;

– протоколи безпеки.

DSA є надійним і добре вивченим алгоритмом для створення та перевірки цифрових підписів, його безпека базується на складності задачі дискретного логарифмування. Хоча він має свої особливості та відмінності від інших

алгоритмів цифрових підписів, він залишається важливим інструментом для забезпечення безпеки та автентичності цифрової інформації.

Гібридне шифрування — це шифрування яке поєднує переваги симетричного та асиметричного шифрування.

Принцип роботи:

– відправник генерує випадковий симетричний ключ;

– дані шифруються за допомогою симетричного алгоритму;

– симетричний ключ шифрується публічним ключем отримувача;

– отримувач використовує свій приватний ключ для дешифрування симетричного ключа, а потім розшифровує дані.

Приклади використання:

– SSL/TLS

– PGP

Переваги використання:

– висока швидкість обробки даних завдяки симетричному шифруванню;

– безпечний обмін ключами через асиметричне шифрування.

Алгоритм хешування — перетворення вхідних даних будь-якої довжини у фіксований розмір хеш-значення, широко використовується для перевірки цілісності даних та зберігання паролів.

Основні алгоритми:

– SHA-2;

– SHA-3;

– MD5.

Приклади застосування:

– перевірка цілісності файлів;

– зберігання паролів у базах даних;

– цифрові підписи.

Алгоритм цифрового підпису — забезпечує автентичність та цілісність повідомлень або документів, створюється за допомогою приватного ключа відправника і може бути перевірений публічним ключем (рисунок 2.12).

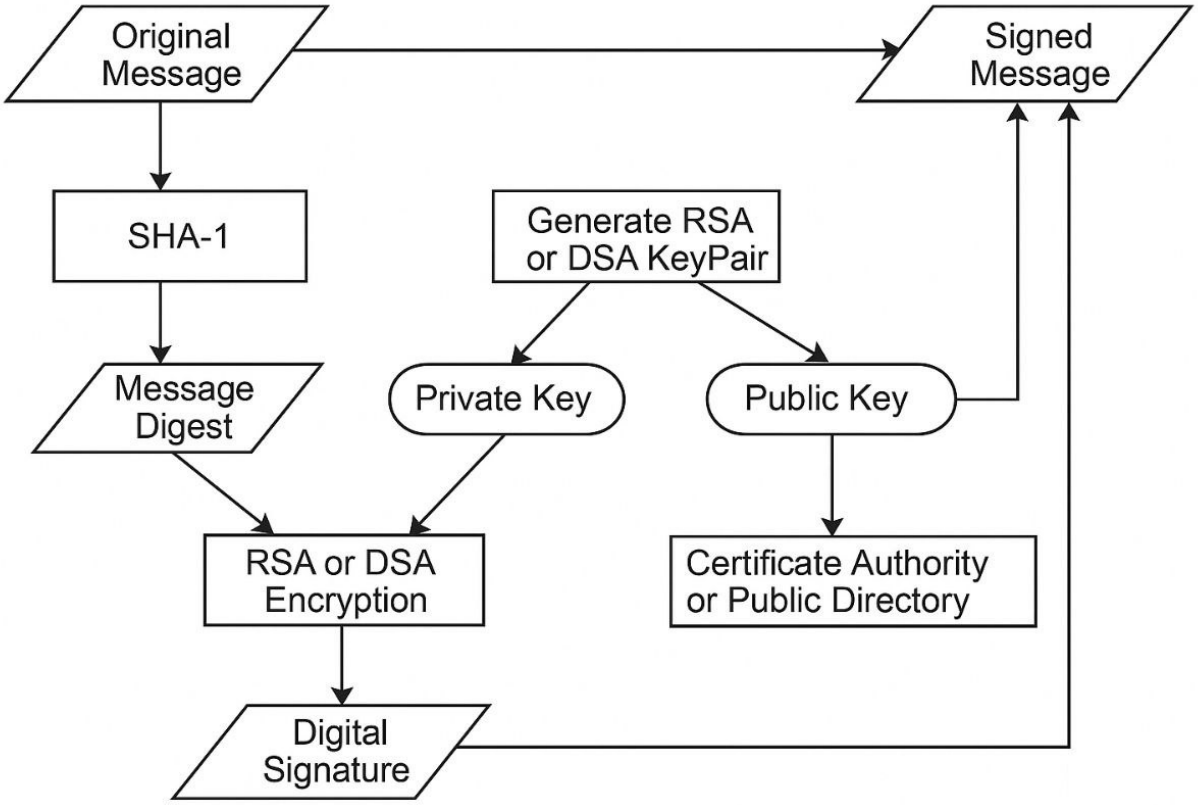


Рисунок 2.12 – Схема створення цифрового підпису

Основні алгоритми:

– RSA;

– DSA;

– ECDSA.

Приклади застосування:

– електронні документи та контракти;

– програмне забезпечення та оновлення;

– системи електронного голосування.

Проведемо порівняння рівнів захисту (таблиця 2.1).

Таблиця 2.1 – Порівняння рівнів захисту

Рівень захисту	Алгоритми	Переваги	Недоліки
Симетричне шифрування	AES, DES	Висока швидкість, ефективність	Потребує безпечного обміну ключами
Асиметричне шифрування	RSA, ECC	Безпечний обмін ключами, цифрові підписи	Нижча швидкість, більші розміри ключів
Гібридне шифрування	AES, RSA/ECC	Поєднання двох видів шифрування	Складність реалізації
Подвійне шифрування	AES, AES/RSA	Додатковий рівень захисту	Зниження продуктивності
Шифрування файлової системи	LUKS, Bitlocker	Прозорість для користувача, захист всього диска	Може бути вразливим до атак при завантаженні

2.2 Класифікація файлів за рівнем таємності

Класифікація файлів за рівнем таємності — це процес категоризації інформації, що міститься у файлах, відповідно до ступеня її чутливості та потенційної шкоди, яка може бути завдана внаслідок її несанкціонованого розголошення, модифікації або знищення. Ця класифікація є основою для застосування відповідних заходів безпеки, спрямованих на захист інформації протягом усього її життєвого циклу.

Призначення класифікації:

– запобігти несанкціонованому доступу до критично важливих даних;

– оптимізувати витрати на захист;

– визначити відповідний рівень шифрування та контроль доступу.

Основні рівні таємності інформації:

– несекретна;

– службова;

– конфіденційна;

– цілком таємна;

Класифікація файлів за рівнем таємності наведена в таблиці 2.2.

Таблиця 2.2 – Порівняння рівнів захисту

Рівень таємності	Ознаки	Документи
1	2	3
Публічний	Відкритий доступ Не містить критичної інформації Не вимагає захисту, але може мати обмеження на редагування	Рекламні матеріали Публічні фінансові звіти Веб-контент Прес-релізи Інструкції клієнтів
Внутрішній	Не призначений для публічного доступу Може бути опрацьований усіма співробітниками організації	Політики компанії Внутрішні презентації Робочі інструкції Організаційні листи Технічні специфікації внутрішніх продуктів
Конфіденційний	Обмежений доступ Розголошення може вплинути на репутацію або порушити нормативні вимоги Потрібна автентифікація та базове шифрування	Персональні дані Медичні записи Контракти з партнерами Звіти про внутрішній аудит Протоколи зустрічей управління

Кінець таблиці 2.2

1	2	3
Внутрішній	Не призначений для публічного доступу Може бути опрацьований усіма співробітниками організації Захист обмежений	Політики компанії Внутрішні презентації Робочі інструкції Організаційні листи Технічні специфікації внутрішніх продуктів

Несекретна інформація — це інформація, яка не підпадає під жодну категорію обмеженого доступу або конфіденційності, має мінімальні наслідки у разі витоку, може потребувати певних заходів безпеки для забезпечення її цілісності та доступності.

Приклад відкритої інформації:

– внутрішні інструкції користування технікою;

– загальні плани навчання.

Службова інформація — це документи та дані, які використовуються в процесі виконання службових обов'язків співробітниками організації, має середній рівень захисту.

Приклади службової інформації:

– службові записи;

– внутрішні накази;

– робочі файли;

Заходи безпеки для файлів службової інформації зазвичай спрямовані на забезпечення їхньої доступності, цілісності та керованості. Можуть застосовуватися такі заходи, як контроль доступу на рівні файлової системи, резервне копіювання, антивірусний захист та базові правила кібергігієни для співробітників.

Файли службової інформації є важливим елементом внутрішньої комунікації та забезпечення робочих процесів в організації, не вимагаючи при цьому суворих заходів безпеки, як конфіденційні або секретні дані.

Конфіденційна інформація — це дані, доступ до яких обмежено певним колом осіб, несанкціоноване розголошення, модифікація або знищення якої може призвести до негативних наслідків для організації, фізичних осіб або держави.

Доступ до цих файлів надається лише авторизованим співробітникам. Для забезпечення конфіденційності таких файлів застосовується високий рівень захисту.

Приклади конфіденційної інформації:

– дані пацієнтів;

– паспорти;

– паролі та ключі шифрування;

– юридична документація.

Захист файлів конфіденційної інформації є критично важливим для запобігання фінансовим втратам, репутаційним ризикам, юридичним наслідкам та порушенням приватності.

Цілком таємна інформація — це інформація, доступ до якої суворо обмежений через її чутливість та потенційну загрозу національній безпеці, обороні, зовнішній політиці, економіці або іншим життєво важливим інтересам держави, для цього використовують максимально високий рівень захисту, використовуються комплексні заходи безпеки, спрямовані на запобігання будь-якому несанкціонованому доступу та розголошенню. Застосовуються жорстко регламентовані процедури створення, обробки, копіювання, передачі, зберігання та знищення таких файлів, що часто включають спеціальні приміщення, обладнання та протоколи.

Приклади цілком таємної інформації:

– плани стратегічних військових операцій;

– інформація про важливі розвідувальні джерела та методи їх обробки;

– ключі стратегічного управління;

– деталі озброєнь.

Забезпечення безпеки файлів цілком таємної інформації є найвищим пріоритетом для держави. Для цього створюються спеціалізовані системи захисту, які включають організаційні, технічні та фізичні заходи безпеки, а також ретельний відбір та підготовку персоналу, який має доступ до цієї інформації.

У програмному забезпеченні, яке реалізує систему шифрування, кожен файл або документ перед шифруванням отримує "мітку рівня таємності", на основі якої система виконує наступні дії:

– автоматично застосовує відповідний алгоритм шифрування;

– визначає, хто має право на перегляд або редагування файлу;

– фіксує дії над такими файлами у журналі безпеки.

2.3 Вибір алгоритмів шифрування відповідно до рівня таємності

Шифрування є ключовим інструментом захисту інформації. Однак ефективність шифрування залежить не лише від самого факту його використання, а й від правильно обраного алгоритму відповідно до рівня таємності даних.

Застосування дуже слабких алгоритмів створює ризики, а надсильних може ускладнювати обробку та уповільнювати роботу системи. Саме тому важливо реалізувати гнучкий підхід до вибору методів шифрування.

Основні критерії вибору алгоритму:

– можливість швидкого дешифрування;

– рівень таємності інформації;

– продуктивність системи;

– наявність сертифікацій або стандарту;

– сумісність з іншими системами.

Для несекретного рівня таємності можна використовувати Base64 або XOR, ці методи не забезпечать криптографічний захист, однак перешкоджають моментальному прочитання вмісту у разі несанкціонованого доступу.

Переваги:

– швидкість;

– мінімальні обчислення.

Недоліки:

– легко зламати;

– не підходить для чутливої інформації.

Для службової інформації рекомендовано використати AES-128, він використовує алгоритм симетричного блокового шифрування, який забезпечить високий рівень безпеки.

Переваги:

– оптимальне співвідношення швидкості та безпеки;

– достатній рівень захисту для інформації середньої чутливості.

Вимагає захищеного зберігання або передавання ключа.

Для конфіденційної інформації порівняно з службовою потрібний сильніший рівень захисту, для досягнення мети рекомендовано використовувати AES-256, який забезпечить високий рівень захисту навіть від потужних атак або гібридне шифрування AES/RSA, яке комбінує швидкість симетричного шифрування AES з безпечним обміном ключами за допомогою асиметричного RSA.

Переваги використання саме цих алгоритмів:

– стійкість до атак;

– відповідність сучасним стандартам безпеки.

Недоліки покращення захисту:

– вимагає більшої обчислювальної потужності;

– складніша система управління ключами.

Цілком таємна інформація має стратегічне значення для держави, втрата, пошкодження або пошкодження може призвести до серйозних наслідків, тому для даного рівня вибираєм AES-256 з режимом GCM або XTS, де поєднується найнадійніше програмне шифрування з апаратними рішеннями.

Додатково впроваджуються такі заходи:

– контроль доступу;

– журнали аудиту;

– багатофакторна автентифікація.

Раціональний вибір криптографічних алгоритмів залежно від рівня таємності дозволяє досягти балансу між безпекою, продуктивністю та зручністю користування. Такий підхід забезпечує високий рівень захисту критичної інформації без перевантаження системи чи зайвих витрат.

2.4 Механізми контролю доступу та керування ключами шифрування

У контексті захисту конфіденційної інформації, особливо при її шифруванні, критично важливу роль відіграють механізми контролю доступу та ефективне керування криптографічними ключами. Ці два аспекти тісно пов'язані між собою та є фундаментом для забезпечення безпеки зашифрованих даних. Недостатньо надійно зашифрувати інформацію, якщо не контролюється доступ до неї та до ключів, необхідних для її дешифрування. У цьому розділі буде детально розглянуто основні механізми контролю доступу до зашифрованих даних та ключових матеріалів, а також методи їхнього безпечного керування протягом усього життєвого циклу.

Контроль доступу до зашифрованих даних є багаторівневим процесом, спрямованим на те, щоб лише авторизовані суб'єкти мали можливість отримати доступ до інформації у зрозумілому вигляді.

Основні механізми контролю доступу:

– ідентифікація та автентифікація суб’єктів;

– авторизація суб’єктів доступу;

– контроль доступу на рівні даних.

Першим кроком у контролі доступу є ідентифікація суб'єкта, який запитує доступ до зашифрованих даних. Ідентифікація передбачає надання суб'єктом

унікального ідентифікатора. Після ідентифікації відбувається процес підтвердження заявленої ідентичності суб'єкта. Для автентифікації можуть використовуватися різні методи:

– паролі;

– багатофакторна автентифікація, яка вимагає використання двох або більше різних факторів;

– використання унікальних біологічних характеристик користувача для автентифікації;

– цифрові сертифікати;

– апаратні токени.

Вибір методів автентифікації залежить від рівня чутливості зашифрованих даних та потенційних загроз. Для високочутливої інформації потрібно використовувати багаторівневі та надійні методи автентифікації.

Після успішної автентифікації відбувається процес авторизації, який визначає, які дії автентифікований суб'єкт має право виконувати з зашифрованими даними. Механізми авторизації можуть базуватися на таких моделях як:

– дискреційний контроль доступу;

– мандатний контроль доступу;

– контроль доступу на основі ролей;

– контроль доступу на основі атрибутів.

Дискреційний контроль доступу — власник об'єкта визначає, хто має право доступу та які саме дії вони можуть виконувати.

Мандатний контроль доступу — система визначає права доступу на основі рівня таємності об'єктів та формального допуску суб'єктів. Користувачі не можуть самостійно змінювати права доступу.

Контроль доступу на основі ролей — права доступу призначаються ролям, а користувачі призначаються на ці ролі, що спрощує адміністрування прав доступу у великих системах.

Контроль доступу на основі атрибутів — права доступу визначаються на основі набору атрибутів суб'єкта, об'єкта та контексту запиту. Модель ABAC є найбільш гнучкою, але й найскладнішою в реалізації.

Для зашифрованих даних авторизація часто визначає, чи має користувач право на отримання ключа дешифрування для певного файлу або групи файлів.

Окрім контролю доступу до файлів як об'єктів, можуть застосовуватися механізми контролю доступу на рівні самих даних. Це особливо актуально у випадках, коли в одному файлі міститься інформація різної чутливості, такі механізми включають:

– шифрування окремих полів або записів у базі даних різними ключами з різними політиками доступу;

– використання маскування даних або анонімізація для надання доступу до менш чутливих частин інформації.

Ефективне керування ключами шифрування є критично важливим для забезпечення безпеки зашифрованих даних. Компрометація ключів робить безглуздим навіть найсильніший алгоритм шифрування. Процес керування ключами охоплює всі етапи їхнього життєвого циклу:

– генерацію;

– зберігання;

– розповсюдження;

– використання;

– ротацію та знищення.

Ключі шифрування повинні бути криптографічно сильними, тобто мати достатню довжину та бути згенеровані за допомогою криптографічно безпечних генераторів випадкових чисел. Використання слабких або передбачуваних ключів значно полегшує їхнє розкриття.

Зберігання ключів є одним із найважливіших аспектів керування ними. Ключі повинні бути захищені від несанкціонованого доступу, копіювання та модифікації. Для цього можуть використовуватися різні методи, такі як:

– апаратні модулі безпеки;

– криптоконтейнери;

– системи керування ключами;

– довірене оточення;

– шифрування ключів.

Апаратні модулі безпеки — це спеціалізовані апаратні пристрої, розроблені для безпечного зберігання та управління криптографічними ключами. HSM забезпечують високий рівень фізичного та логічного захисту.

Криптоконтейнери — це захищені програмні сховища, які шифрують ключі за допомогою майстер-пароля або іншого ключа.

Системи керування ключами — це централізовані системи для управління ключами шифрування в масштабі підприємства, вони забезпечують централізоване зберігання, розподіл, ротацію та аудит використання ключів.

Довірене виконання оточення включають являють собою захищені області в головному процесорі, які забезпечують ізольоване виконання чутливого коду та зберігання ключів.

Вибір методу зберігання ключів залежить від вимог безпеки, вартості та складності інфраструктури.

Розповсюдження ключів між авторизованими суб'єктами повинно здійснюватися безпечними каналами зв'язку, щоб запобігти їхньому перехопленню. Методи розповсюдження:

Використання криптографічних протоколів, таких як TLS/SSL або SSH, які забезпечують зашифрований канал для передачі ключів;

Попереднє узгодження, коли ключі заздалегідь узгоджуються між сторонами, що обмінюються даними.

Протоколи обміну ключами — дозволяють двом сторонам спільно створити спільний секретний ключ через незахищений канал.

Асиметричне шифрування, коли відправник шифрує симетричний ключ за допомогою відкритого ключа одержувача. Лише одержувач, володіючи відповідним приватним ключем, може розшифрувати симетричний ключ.

Процес використання ключів для шифрування та дешифрування даних повинен бути надійно інтегрований у відповідні системи та програми. Важливо мінімізувати час знаходження ключів у пам'яті та запобігати їхньому випадковому розголошенню, можна використовувати дампи пам'яті або файли підкачки.

Також періодична зміна криптографічних ключів є важливою практикою безпеки. Ротація обмежує кількість даних, скомпрометованих у випадку розкриття ключа, та знижує ймовірність успішних атак, спрямованих на тривале використання одного й того самого ключа.

Коли ключ більше не потрібен, він повинен бути безпечно знищений, щоб запобігти його відновленню та подальшому несанкціонованому використанню. Просте видалення файлу з ключем є недостатнім, потрібно використовувати спеціальні методи безповоротного видалення даних, такі як перезапис носіїв інформації випадковими даними. Для апаратних модулів безпеки існують спеціальні команди для безпечного знищення ключів.

Важливо розуміти, що механізми контролю доступу та керування ключами шифрування не є незалежними, вони тісно взаємодіють для забезпечення комплексного захисту зашифрованих даних. Для отримання доступу до зашифрованого файлу користувач повинен пройти автентифікацію та бути авторизованим для отримання ключа дешифрування для цього файлу. Система керування ключами повинна забезпечити безпечне зберігання та надання цього ключа лише авторизованим користувачам.

Ефективна інтеграція механізмів контролю доступу та керування ключами є запорукою надійної системи захисту зашифрованої інформації.

3 ТЕХНОЛОГІЇ ТА АЛГОРИТМИ ШИФРУВАННЯ ФАЙЛІВ

3.1 Основні компоненти системи шифрування з багаторівневим захистом

Першим кроком є визначення рівнів чутливості даних та відповідних рівнів доступу. Сучасні установи, як-от державні органи, медичні заклади чи приватні компанії, оперують великою кількістю конфіденційної інформації, що потребує надійного захисту. Для забезпечення безпеки зберігання та передачі таких даних була реалізована система шифрування файлів з різними рівнями захисту. У межах цієї системи файли класифікуються відповідно до їхньої важливості та ступеня чутливості, після чого до них застосовується відповідний алгоритм шифрування.

Основні типи даних:

– загальнодоступні дані;

– конфіденційні дані;

– секретні дані;

– критично важливі дані.

Загальнодоступні дані — це інформація, яка не є конфіденційною, не захищена авторським правом або іншими обмеженнями доступу, і тому може вільно використовуватися, поширюватися та модифікуватися будь-ким без будь-яких дозволів чи ліцензій.

Ключові характеристики:

– немає юридичних чи технічних перешкод для доступу, використання та поширення цих даних;

– будь-хто може використовувати ці дані для будь-якої мети, комерційної чи некомерційної;

– відсутність авторського права;

– державні дані, часто є загальнодоступними, за винятком, коли вони становлять державну таємницю або містять особисту інформацію.

Приклади даних, які можуть бути загальнодоступними:

– статичні дані, опубліковані урядовим агенствами;

– закони та нормативні акти;

– рішення судів;

– наукові дослідження, опубліковані у відкритому доступі;

– книги та музика, термін дії авторського права на які минув;

– деякі географічні дані та карти;

– інформація, добровільно передана у суспільне надбання її автором.

Конфіденційні дані — це інформація, доступ до якої обмежено певним колом осіб через її чутливий характер або юридичні вимоги. Розкриття, несанкціонований доступ або використання таких даних може призвести до негативних наслідків для фізичних осіб, організацій або держави.

Ключові характеристики:

– доступ до цих даних надається лише авторизованим особам або системам;

– розкриття цієї інформації може завдати шкоди;

– часто конфіденційні дані захищені законами та нормативними актами;

– потреба в захисті.

Приклади даних, які можуть бути конфіденційними:

– персональні дані ;

– комерційна таємниця;

– державна таємниця;

– юридична інформація;

– паролі та ключі шифрування.

Секретні дані — це підкатегорія конфіденційних даних, яка характеризується особливо високим рівнем чутливості та необхідністю надзвичайно суворого контролю доступу. Їх розкриття може завдати винятково серйозної шкоди національній безпеці, обороні, зовнішній політиці, економіці або іншим життєво важливим інтересам держави чи установи.

Ключові характеристики:

– інформація має стратегічне значення і її розкриття може мати катастрофічні наслідки;

– найсуворіші заходи безпеки для запобігання несанкціонованому доступу, копіюванню, розголошенню або знищенню;

– юридична відповідальність;

– чітко визначені процедури.

Приклади даних, які можуть бути секретними:

– державна таємниця найвищого рівня;

– комерційна таємниця;

– інформація про спеціальні операції правоохоронних органів.

Критично важливі дані — це інформація, яка є абсолютно необхідною для безперебійного функціонування ключових бізнес-процесів, систем або операцій організації. Втрата таких пошкодження або недоступність таких даних може призвести до серйозних наслідків.

Ключові характеристики:

– незамінність для ключових процесів;

– високий рівень впливу;

– потреба у високій доступності та цілісності;

– потреба в резервному копіюванні та відновлення.

Приклади даних, які можуть бути критично важливими:

– фінансові дані;

– дані клієнтів;

– виробничі дані;

– логістичні дані;

– медична інформація;

– дані про безпеку польотів;

– операційні дані енергетичних систем.

Наступним кроком є вибір криптографічних алгоритмів, наприклад таких як:

– симетричне шифрування, яке забезпечує високу швидкість та ефективність для великих обсягів даних;

– асиметричне шифрування, яке використовують для захисту ключів шифрування та цифрових підписів;

– гібридні методи, які поєднують в собі безпеку та швидкість симетричних та асиметричних алгоритмів.

Далі потрібно вибрати механізм управління ключами, так як управління криптографічними ключами є критично важливо.

Приклади управління ключами:

– використання надійних генераторів випадкових чисел;

– використання апаратних модулів або захищених сховищ для зберігання ключів;

– регулярна зміна ключів для мінімізації ризику компрометації.

Система шифрування повинна бути інтегрована з механізмами автентифікації та авторизації, тому потрібно використовувати наступні механізми:

– використання багатофакторної автентифікації;

– розмежування доступу.

Для забезпечення прозорості дій та виявлення потенційних загроз необхідно впровадити:

– фіксацію всіх операцій з файлами та ключами;

– використання системи виявлення аномалій для виявлення підозрілих дій.

Реалізація системи шифрування файлів з різними рівнями захисту є складним, але необхідним завданням для забезпечення конфіденційності, цілісності та доступності інформації.

Поєднання різних криптографічних методів, ефективне управління ключами та інтеграція з системами контролю доступу дозволяють створити гнучку та надійну систему захисту даних.

3.2 Модель безпеки шифрування файлів

Модель безпеки шифрування файлів — це комплекс принципів, технологій, процедур і практик, спрямованих на захист інформації, що зберігається у файлах, шляхом її перетворення у нечитабельний формат і забезпечення контролю над доступом до ключів дешифрування. Головною метою є гарантія конфіденційності даних, цілісність та доступність, простіше кажучи, це структурований підхід до того, як ми застосовуємо шифрування для захисту файлів, включаючи все, починаючи від вибору алгоритму шифрування і закінчуючи тим, як ми керуємо ключами, хто має доступ до зашифрованих даних і як ми реагуємо на можливі загрози (рисунок 3.1).

Рисунок 3.1 – Модулі системи шифрування

Без розробленої моделі безпеки шифрування будь-яка реалізація системи буде фрагментарною, вразливою до внутрішніх і зовнішніх загроз.

Модель безпеки шифрування файлів включає в себе:

– шифрування даних;

– створення надійних криптографічних ключів, безпечне зберігання ключів шифрування та дешифрування, безпечний обмін ключами та їх безпечне видалення;

– перевірка особистості користувача або системи, яка намагається отримати доступ до даних, визначення прав доступу автентифікованих користувачів до цих даних;

– захист від несанкціонованого доступу;

– забезпечення цілісності даних;

– аудит та журнали;

– процедури відновлення;

– управління політиками безпеки.

Сучасні організації, які оперують конфіденційною або особливо важливою інформацією, не можуть обійтись без комплексної моделі захисту, оскільки вона є основою інформаційної безпеки в умовах цифрової трансформації.

Модель передбачає, що кожен користувач має бути унікально ідентифікований та пройти процедуру автентифікації перед отриманням доступу до системи. Методи автентифікації включають:

– біометричні методи;

– токени доступу;

– логін/пароль;

– двофакторна автентифікація.

Файли розподіляються за категоріями відповідно до їхньої критичності.

Основні категорії:

– відкрита інформація;

– службова інформація;

– конфіденційна;

– цілком таємна.

Модель передбачає автоматизований вибір алгоритму на основі рівня чутливості:

– симетричне шифрування;

– асиметричне шифрування;

– гібридні системи.

Система шифрування повинна мати вбудований механізм генерації, збереження, розповсюдження, ротації та відкликання ключів. Для цього використовуються:

– внутрішні сервіси, такі як модулі в ОС або спеціалізовані програми;

– зовнішні сховища;

– апаратні рішення.

Модель регламентує, що користувачі мають доступ лише до тих файлів, які відповідають їх ролі в системі (Role-Based Access Control). Можуть бути реалізовані також моделі MAC (Mandatory Access Control) та DAC (Discretionary Access Control).

Кожна дія над шифрованими файлами повинна записуватись у захищений журнал. Це дозволяє:

– виявляти підозрілі дії;

– проводити службові розслідування;

– формувати звіти для аудитів безпеки.

Також модель повинна містити чіткий план дій у разі виявлення витоку або спроби несанкціонованого доступу:

– негайне відкликання ключів;

– блокування користувача;

– повідомлення адміністраторів;

– відновлення з резервної копії.

При реалізація моделі безпеки можна стикнутись з такими труднощами:

– людський фактор;

– висока складність управління при великій кількості користувачів та ключів;

– ризики втрати ключів або компрометації;

– необхідність балансування між зручністю та рівнем захисту.

У перспективі моделі безпеки шифрування розвиватимуться у напрямках:

– квантово-стійких алгоритмів;

– інтеграції з Zero Trust моделлю;

– повної автоматизації керування ключами на базі AI.

Модель безпеки шифрування є стратегічною основою для ефективного захисту файлів. Її впровадження дозволяє організації не тільки уникнути витоку конфіденційних даних, але й забезпечити юридичну відповідність. В умовах зростання цифрових загроз модель має бути не статичною схемою, а динамічним інструментом, що постійно адаптується до нових викликів.

3.3 Реалізація системи шифрування файлів

У контексті забезпечення інформаційної безпеки установи важливим є впровадження системи шифрування файлів, яка б враховувала різні рівні чутливості інформації. Цей розділ присвячено розробці такої системи, зосереджуючись на рівні захисту, призначеному для внутрішніх службових документів.

Розпочнемо з вибору рівня захисту інформації. Внутрішні службові документи в установі, як правило, містять інформацію, яка не є публічною, але її розголошення може призвести до певних негативних наслідків, таких як порушення внутрішніх процесів, несанкціонований доступ до службової інформації, або репутаційні ризики. Враховуючи це, для захисту таких документів доцільно обрати симетричне шифрування з використанням надійного алгоритму та достатньо довгим ключем. Симетричне шифрування забезпечить високу швидкість шифрування та дешифрування, що є важливим при роботі з великою

кількістю внутрішніх документів. Алгоритм AES на сьогодні є одним з найбільш надійних та широко використовуваних алгоритмів. Для службових документів доцільно використовувати AES з довжиною ключа 256 біт, який забезпечить високий рівень криптографічної стійкості.

Отож, перейдемо до реалізації системи шифрування. Для кожного файлу або групи файлів буде розроблена генерація ключа де буде генеруватись унікальний випадковий симетричний ключ.

Реалізація буде проводитись мовою python, перед тим як користувач зможе виконувати будь-які операції, він повинен пройти автентифікацію, це реалізовано у функції блок-схема якої показана на рисунку 3.2.

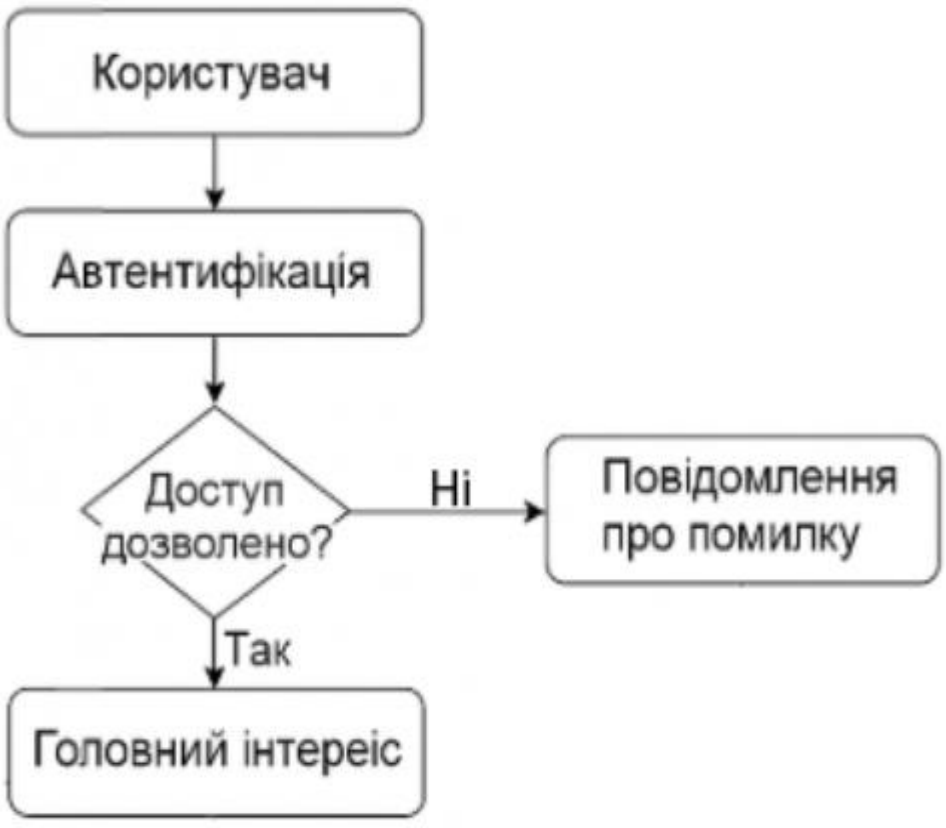


Рисунок 3.2 – Логіка автентифікації користувача

Вікно автентифікації відображено на рисунку 3.3.

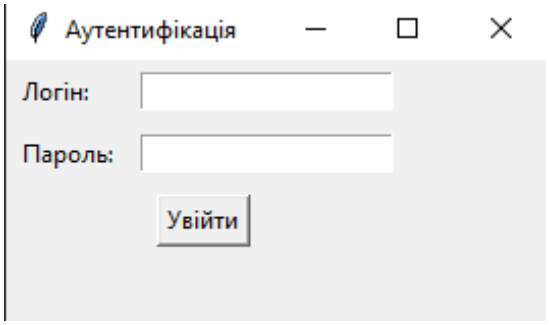


Рисунок 3.3 – Вікно автентифікація користувача

Після успішної автентифікації створюється основний графічний інтерфейс, де користувач може вибирати файли, вводити пароль та обирати алгоритм, це реалізовано у функції блок-схема якої показана на рисунку 3.4.

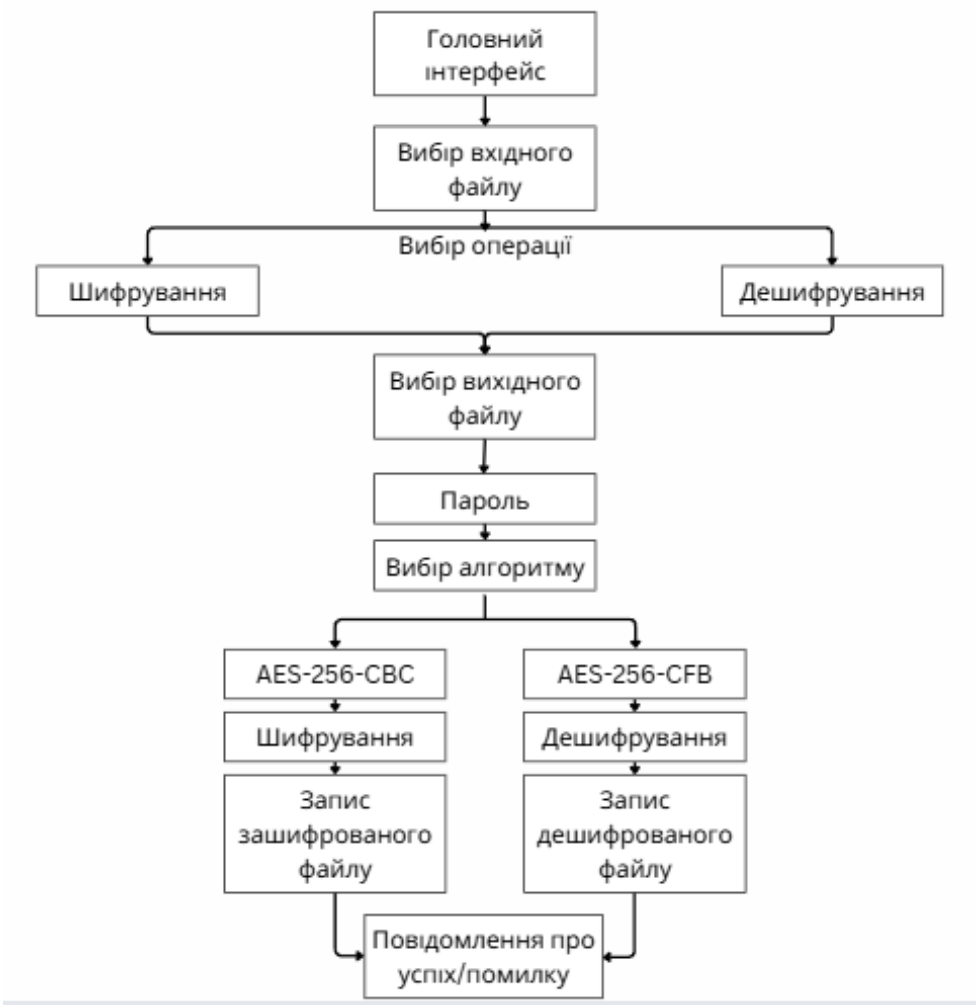


Рисунок 3.4 – Блок-схема взаємодії з інтерфейсом

Вікно головного інтерфейсу відображено на рисунку 3.5.

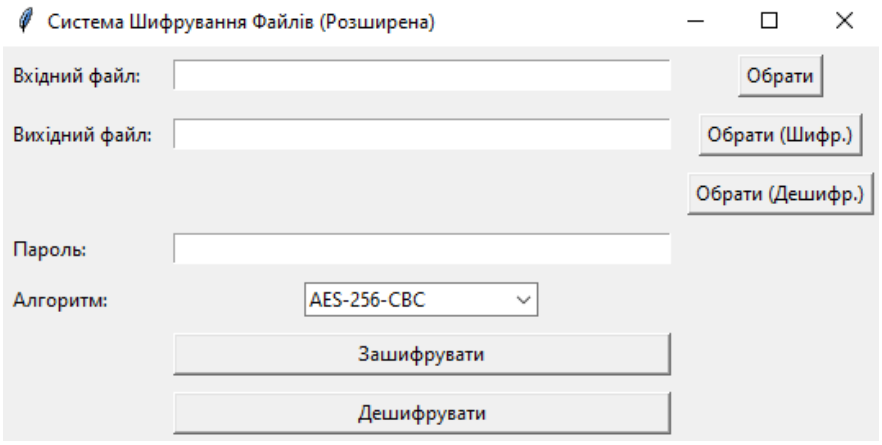


Рисунок 3.5 – Головний інтерфейс

Користувач використовує кнопки для вибору файлів, які потрібно зашифрувати або дешифрувати, також для визначення шляху вихідного файлу, це реалізовано у функції зображеній на рисунку 3.6.

```
def select_input_file():
def select_output_file(operation):
```

Рисунок 3.6 – Вибір вхідного та вихідного файлів

Введений пароль шифрування використовується для генерації криптографічного ключа. Для підвищення безпеки використовується сіль та функція PBKDF2HMAC, це реалізовано у функції зображеній на рисунку 3.7.

```
def generate_derived_key(password: str, salt: bytes) -> bytes:
    password_bytes = password.encode('utf-8')
    kdf = PBKDF2HMAC()
    return kdf.derive(password_bytes)
```

Рисунок 3.7 – Генерація ключа шифрування на основі пароля

Після вибору файлів, введення пароля для шифрування та вибору алгоритму, при натисканні кнопки "Зашифрувати" викликається функція perform_encryption, яка, в свою чергу, викликає encrypt_file, ця функція зчитує

вхідний файл частинами, застосовує обраний алгоритм шифрування з використанням згенерованого ключа та випадкового вектора ініціалізації і записує зашифровані дані у вихідний файл. Для AES-256-CBC застосовується padding, на початку зашифрованого файлу записується назва алгоритму, сіль та вектор, це реалізовано у функції зображеній на рисунку 3.8-3.9.

Рисунок 3.8 – Шифрування файлу

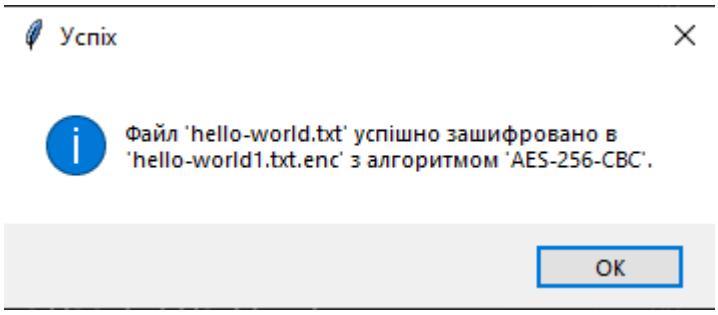


Рисунок 3.9 – Вікно успішного шифрування файлу

При натисканні кнопки "Дешифрувати", залежно від зчитаного алгоритму, застосовується відповідний метод дешифрування, це реалізовано у функції зображеній на рисунку 3.10, вікно успішного дешифрування відображено на рисунку 3.11.

```
def decrypt_file(input_file: str, output_file: str, password: str):
```

Рисунок 3.10 – Дешифрування файлу

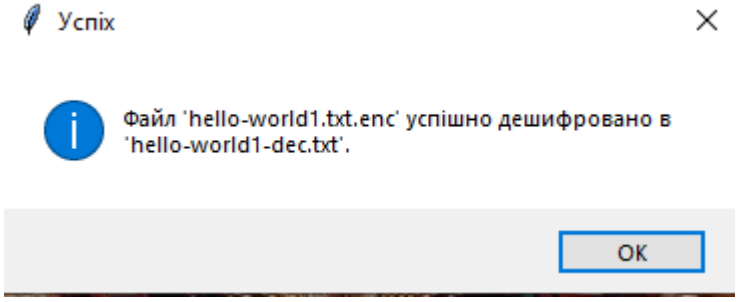


Рисунок 3.11 – Вікно успішного дешифрування файлу

Опис блоку дешифрування:

– початок виконання функції decrypt_file;

– перевірка, чи користувач автентифікований verify_token;

– виведення повідомлення про необхідність автентифікації у разі помилки;

– відкриття файлу, який потрібно дешифрувати;

– читання першого рядка файлу для визначення алгоритму шифрування;

– читання випадкової солі, яка використовувалась при шифруванні;

– читання вектору ініціалізації, який використовується при шифруванні.

– створення ключа дешифрування на основі введеного пароля та зчитаної солі.

Якщо використовувався алгоритм AES-256-CBC:

– перевірка, чи використовувався алгоритм алгоритм AES-256-CBC;

– створення об'єкта Cipther для дешифрування в режимі CBC;

– ініціалізація об'єкта дешифратора;

– створення об'єкта, який видаляє додане при шифруванні padding;

– зчитування всіх зашифрованих даних

– виконання операції дешифрування;

– перевірка, чи не виникала помилка під час видалення padding;

– реєстрація помилки та виведення повідомлення користувачу;

– видалення padding із розшифрованих даних;

– створення або відкриття файлу для запису розшифрованих даних;

– збереження розшифрованого вмісту;

– виведення повідомлення про успішне завершення операції;

Якщо використовувався алгоритм AES-256-CFB:

– перевірка, чи використовувався алгоритм AES-256-CFB;

– створення об'єкта Cipher для дешифрування в режимі CFB;

– поступове зчитування зашифрованих даних частинами;

– дешифрування поточної частини даних;

– збереження розшифрованої частини;

– завершення процесу дешифрування;

– виведення повідомлення про успішне виконання;

ВИСНОВКИ

Під час кваліфікаційної роботи було успішно створено та реалізовано програмну систему для шифрування та дешифрування файлів, що відповідає сучасним вимогам до криптографічного захисту даних. Головною метою дослідження стало розроблення інструменту, що поєднує функціональність, безпеку та зручність у використанні для забезпечення конфіденційності даних. Виконання поставленої задачі стало можливим завдяки ретельному дотриманню основних етапів розробки, що включали концептуальне проєктування, реалізацію та комплексне тестування програмного продукту. Першим етапом стало глибоке вивчення теоретичних основ криптографії. Це включало аналіз симетричних та асиметричних алгоритмів шифрування, методів генерації ключів, принципів роботи векторів ініціалізації та механізмів доповнення. Особливу увагу було приділено розшифруванню роботи блокових шифрів, таких як AES, що є загальновизнаним стандартом для захисту даних. Вивчення різних режимів роботи AES, зокрема Cipher Block Chaining та Cipher Feedback, дозволило обґрунтовано обрати найбільш підходящі для реалізації з точки зору балансу між безпекою та продуктивністю. Розуміння функцій отримання ключа з пароля, зокрема PBKDF2HMAC, стало критично важливим для забезпечення надійності системи проти атак методом перебору по словнику, дозволяючи генерувати стійкі криптографічні ключі з відносно простих паролів, додаючи сіль та збільшуючи кількість ітерацій.

Наступний етап був присвячений проектуванню архітектури програмної системи. На цьому етапі було визначено основні модулі та їх взаємодію. Була розроблена структурна схема системи, яка візуалізує потік даних від автентифікації користувача до виконання операцій шифрування/дешифрування. Ключовим аспектом архітектури стала модульність, що дозволило відокремити логіку криптографічних операцій від інтерфейсу користувача. Також було продумано механізм зберігання ключів, з можливістю розширення (наприклад, інтеграції з системними сховищами ключів або HSM/KMS), хоча в поточній

реалізації було використано тимчасове зберігання в пам'яті для простоти. Важливим елементом проектування стала імплементація системи аудиту всіх значущих подій, таких як успішна автентифікація, спроби шифрування/дешифрування, а також виникнення помилок, що є критично важливим для моніторингу безпеки та діагностики проблем.

Після успішного проєктування архітектури програмної системи розпочалася реалізація програмного коду. Процес взаємодії з користувачем починається з обов'язкової автентифікації, що є першим бар'єром для несанкціонованого доступу до функцій системи. Після успішного входу користувачеві відкривається головне вікно програми, де він може вибрати вхідний та вихідний файли, ввести пароль та обрати криптографічний алгоритм зі списку підтримуваних. Центральне місце в реалізації посіли функції генерації похідного ключа та основні криптографічні операції. Функція "encrypt_file" відповідає за шифрування файлів. Вона включає запис назви алгоритму, солі та вектора ініціалізації на початку зашифрованого файлу, що є критично важливим для коректного дешифрування. Процес шифрування реалізовано з урахуванням обробки великих файлів за допомогою зчитування даних чанками, що дозволяє уникнути надмірного використання оперативної пам'яті. Для алгоритму AES-256-CBC було впроваджено механізм доповнення (padding) PKCS7, а для AES-256-CFB – потокове шифрування без необхідності padding.

Етап дешифрування виявився найбільш вимогливим до точності реалізації, особливо в режимі Cipher Block Chaining. Спочатку зіштовхнувшись із помилками "Invalid padding bytes" при дешифруванні Cipher Block Chaining, що свідчило про некоректну обробку останніх блоків або видалення padding, проблему було успішно локалізовано та вирішено шляхом оптимізації читання даних для дешифрування. Це підкреслило важливість коректної послідовності операцій та ретельної обробки байтів при роботі з криптографічними примітивами. Успішне дешифрування як у режимі Cipher Feedback, так і після виправлення у режимі Cipher Block Chaining, підтвердило коректність реалізації криптографічних алгоритмів.

Четвертий етап включав тестування та налагодження. Систематична перевірка функціональності кожного модуля та інтеграційне тестування дозволили виявити та усунути помилки. Зокрема, інтенсивне тестування дешифрування Cipher Block Chaining з різними розмірами файлів було ключовим для вирішення проблеми з padding. Використання механізму логування суттєво спростило процес налагодження, надаючи детальну інформацію про хід виконання операцій та причини виникнення помилок.

У підсумку, розроблена система демонструє високий рівень функціональності та безпеки для базових потреб шифрування файлів. Вона надає користувачам можливість захистити свої дані за допомогою стійких криптографічних алгоритмів AES-256 у режимах CBC та CFB, використовуючи похідні ключі від паролів. Система є гнучкою, дозволяючи легко додавати нові алгоритми шифрування та методи зберігання ключів у майбутньому. Отриманий досвід та реалізовані підходи можуть слугувати міцною основою для подальшого розвитку проєкту, включаючи розширення функціоналу, оптимізацію продуктивності та інтеграцію з більш складними системами керування ключами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Електронний цифровий підпис. 2021. С. 1. URL: https://www.kmu.gov.ua/usi-pitannya-po-e-poslugam/sho-tak-elektronnij-cifrovij-pidpis-ecp (дата звернення: 8.02.2025).

2. Що таке SSL/TLS. 2022. С. 1. URL: https://vps.ua/blog/ukr/ssl-tls-protocols-details/ (дата звернення: 8.02.2025).

3. Про державну таємницю. 2024. С. 1. URL: https://zakon.rada.gov.ua/laws/main/3855-12 (дата звернення: 10.02.2025).

4. Розголошення державної таємниці та втрата документів, що містять державну таємницю. 2023. С. 1. URL: https://wiki.legalaid.gov.ua/index.php/Розголошення_державної_таємниці_та_втрата_документів%2C_що_містять_державну_таємницю (дата звернення: 18.02.2025).

5. Різниця між RC4 і AES. 2024. С. 1. URL: https://www.geeksforgeeks.org/difference-between-rc4-and-aes/ (дата звернення: 18.02.2025).

6. Основи шифрування. 2023. С. 1. URL: https://hackyourmom.com/pryvatnist/zahoplyuyuchyj-svit-shyfruvannya-rozglyad-typiv-algorytmiv-ta-yih-zastosuvan/ (дата звернення: 18.02.2025).

7. Алгоритм шифрування RSA, види атак на нього. 2023. С. 1. URL: https://dou.ua/forums/topic/43026/ (дата звернення: 18.02.2025).

8. Шифрування. Типи та алгоритми. Що це і який тип шифрування кращий. 2024. С. 1. URL: https://hostkoss.com/b/uk/encryption-types-algorithms/ (дата звернення: 18.02.2025).

9. RSA і ECC, який з них кращий і чому. 2024. С. 1. URL: https://www.reddit.com/r/cryptography/comments/1aouz9u/rsa_vs_ecc_which_one_is_better_and_why/?rdt=35515 (дата звернення: 18.02.2025).

10. Шифрування, типи і алгоритми. Чим відрізняються і де використовуються. 2022. С. 1. URL: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms/ (дата звернення: 22.02.2025).

11. Основи шифрування. 2023. С. 1. URL: https://hackyourmom.com/pryvatnist/zahoplyuyuchyj-svit-shyfruvannya-rozglyad-typiv-algorytmiv-ta-yih-zastosuvan/ (дата звернення: 22.02.2025).

12. Роль шифрування у захисті особистої інформації. 2024. С. 1. URL: https://mindscope.biz.ua/rol-shyfruvannya-u-zahysti-osobystoyi-informacziyi/ (дата звернення: 22.02.2025).

13. Види шифрування інформації. 2025. С. 1. URL: https://ua5.org/protect/395-vidi-shifruvannya-informaciyi.html (дата звернення: 22.02.2025).

14. Що таке шифрування та як воно працює. 2023. С. 1. URL: https://www.kingston.com/ua/blog/data-security/what-is-encryption (дата звернення: 22.02.2025).

15. Що таке керування ідентифікацією та доступом. 2024. С. 1. URL: https://www.issp.training/post/shcho-take-keruvannya-identyfikatsiyeyu-ta-dostupom (дата звернення: 23.02.2025).

16. Безпека AWS. 2023. С. 1. URL: https://itedu.center/ua/blog/devops/7-krashchykh-praktyk-bezpeky-aws-dlia-zakhystu-danykh-v-khmari (дата звернення: 23.02.2025).

17. Типи алгоритмів шифрування, плюси і мінуси кожного з них. 2024. С. 1. URL: https://www.keyfactor.com/education-center/types-of-encryption-algorithms (дата звернення: 25.02.2025).

18. Сучасна криптографія. 2025. С. 1. URL: https://www.sciencedirect.com/topics/computer-science/modern-cryptography (дата звернення: 25.02.2025).

19. Глибоке занурення в типи шифрування. 2024. С. 1. URL: https://www.sealpath.com/blog/types-of-encryption-guide (дата звернення: 25.02.2025).

20. Три основні типи криптографії. 2024. С. 1. URL: https://www.ibm.com/think/topics/cryptography-types (дата звернення: 28.02.2025).

21. Що таке шифрування. 2024. С. 1. URL: https://www.cisco.com/site/us/en/learn/topics/security/what-is-encryption.html (дата звернення: 28.02.2025).

22. Огляд сучасної криптографії на високому рівні. 2025. С. 1. URL: https://muens.io/modern-cryptography-overview (дата звернення: 03.03.2025).

23. Сучасні алгоритми шифрування. Детальний аналіз та перспективи. 2024. С. 1. URL: https://www.sworldjournal.com/index.php/swj/article/view/swj25-00-023/4544 (дата звернення: 03.03.2025).

24. Стратегії шифрування контрольованої некласифікованої інформації. 2024. С. 1. URL: https://continuumgrc.com/uk/encryption-strategies-for-controlled-unclassified-information-cui-in-hybrid-cloud-systems (дата звернення: 03.03.2025).

25. Шифрування файлів. 2023. С. 1. URL: https://learn.microsoft.com/en-us/windows/win32/fileio/file-encryption (дата звернення: 07.03.2025).

26. Що таке шифрування файлів і навіщо це потрібно. 2023. С. 1. URL: https://hackyourmom.com/osvita/open-source-utylit-dlya-shyfruvannya-fajliv-na-bud-yakyj-vypadok-zhyttya (дата звернення: 07.03.2025).

27. Про державну таємницю. 2024. С. 1. URL: https://zakon.rada.gov.ua/laws/main/3855-12 (дата звернення: 10.03.2025).

28. Звід відомостей, що становлять державну таємницю. 2020. С. 1. URL: https://kodeksy.com.ua/pro_derzhavnu_tayemnitsyu/statja-12.htm (дата звернення: 10.03.2025).

29. Методики побудови системи захисту інформації. 2024. С. 1. URL: https://stud.com.ua/179796/informatika/metodiki_pobudovi_sistem_zahistu_informatsiyi (дата звернення: 15.03.2025).

30. Роль та ефективність методу шифрування AES для забезпечення безпеки та захисту інформаційних систем. 2022. С. 1. URL: https://ur.knute.edu.ua/server/api/core/bitstreams/fce66688-f4f6-47e9-b8aa-3ce3b5e96dd7/content (дата звернення: 15.03.2025).

31. Загрози інформаційної безпеки. 2024. С. 1. URL: https://uk.wikipedia.org/wiki/ (дата звернення: 15.03.2025).

32. Моделі безпеки в інформаційній та/або кібербезпеці. 2022. С. 1. URL: https://e-tk.lntu.edu.ua/pluginfile.php/25321/mod_resource/content/1/%D0%A2%D0%95%D0%9C%D0%90%2010.pdf (дата звернення: 17.03.2025).

33. Правові основи забезпечення безпеки інформаційних технологій. 2022. С. 1. URL: https://sites.google.com/view/dcptoinformat/ (дата звернення: 17.03.2025).

34. Модель безпеки це. 2020. С. 1. URL: https://studfile.net/preview/9649825/page:8/ (дата звернення: 17.03.2025).

35. Класифікація джерел загроз інформаційної безпеки. 2022. С. 1. URL: https://vseosvita.ua/lesson/zahrozy-informatsiinoi-bezpeky-ikh-klasyfikatsiia-klasyfikatsiia-dzherel-zahroz-informatsiinoi-bezpeky-323171.html (дата звернення: 25.03.2025).

36. Недостатнє криптографічне сховище. 2023. С. 1. URL: https://cqr.company/ua/web-vulnerabilities/insufficient-cryptographic-storage/ (дата звернення: 28.03.2025).

37. Роль блокчейну в обміні даними. 2023. С. 1. URL: https://scispace.com/papers/a-survey-of-post-quantum-cryptography-start-of-a-new-race-2rph3bc2ui (дата звернення: 16.04.2025).

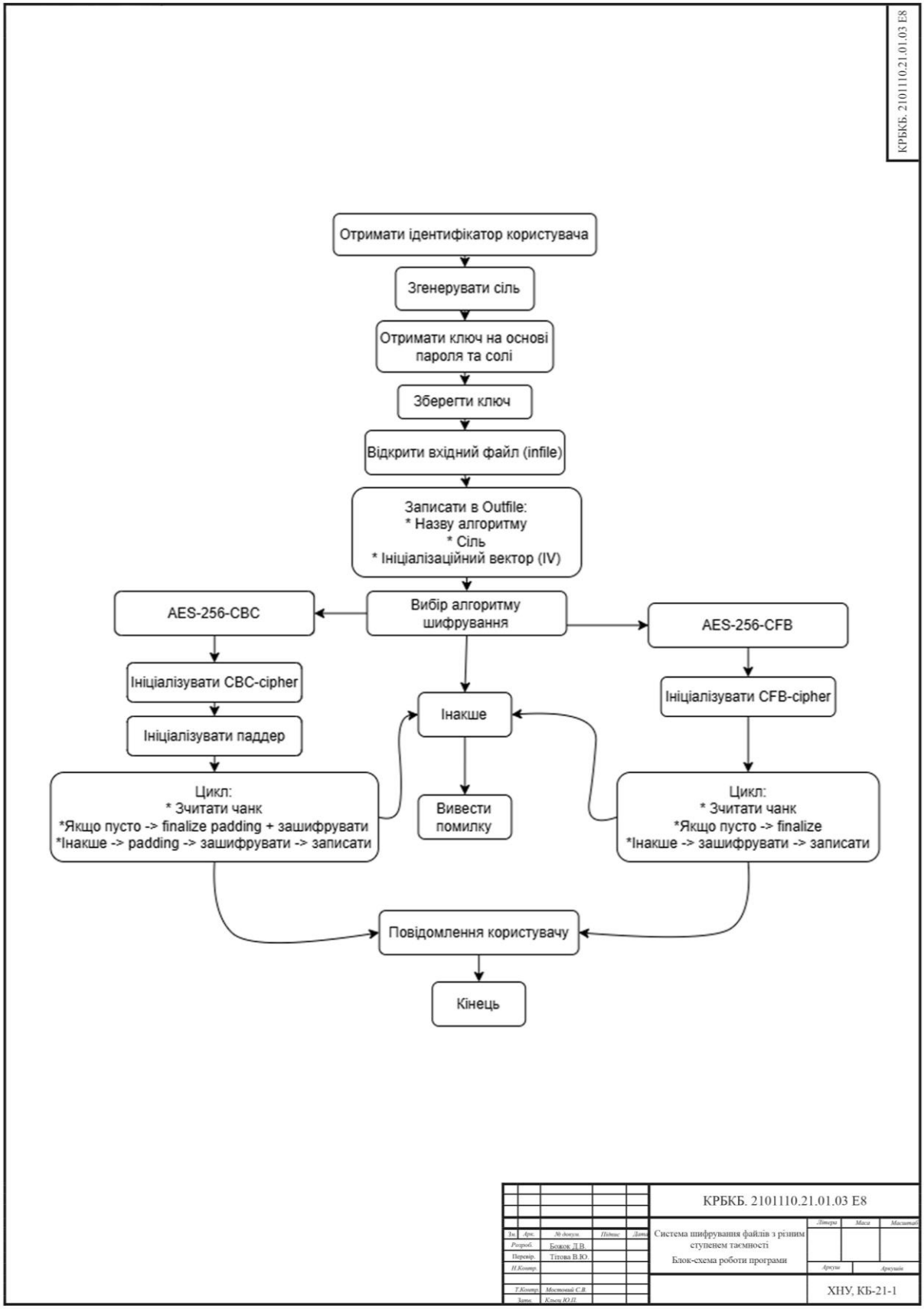
38. PBKDF2HMAC. 2023. С. 1. URL: https://docs.rocketsoftware.com/bundle/unidataunibasiccommands_rg_824/page/vmn1685024690247.html (дата звернення: 20.04.2025).

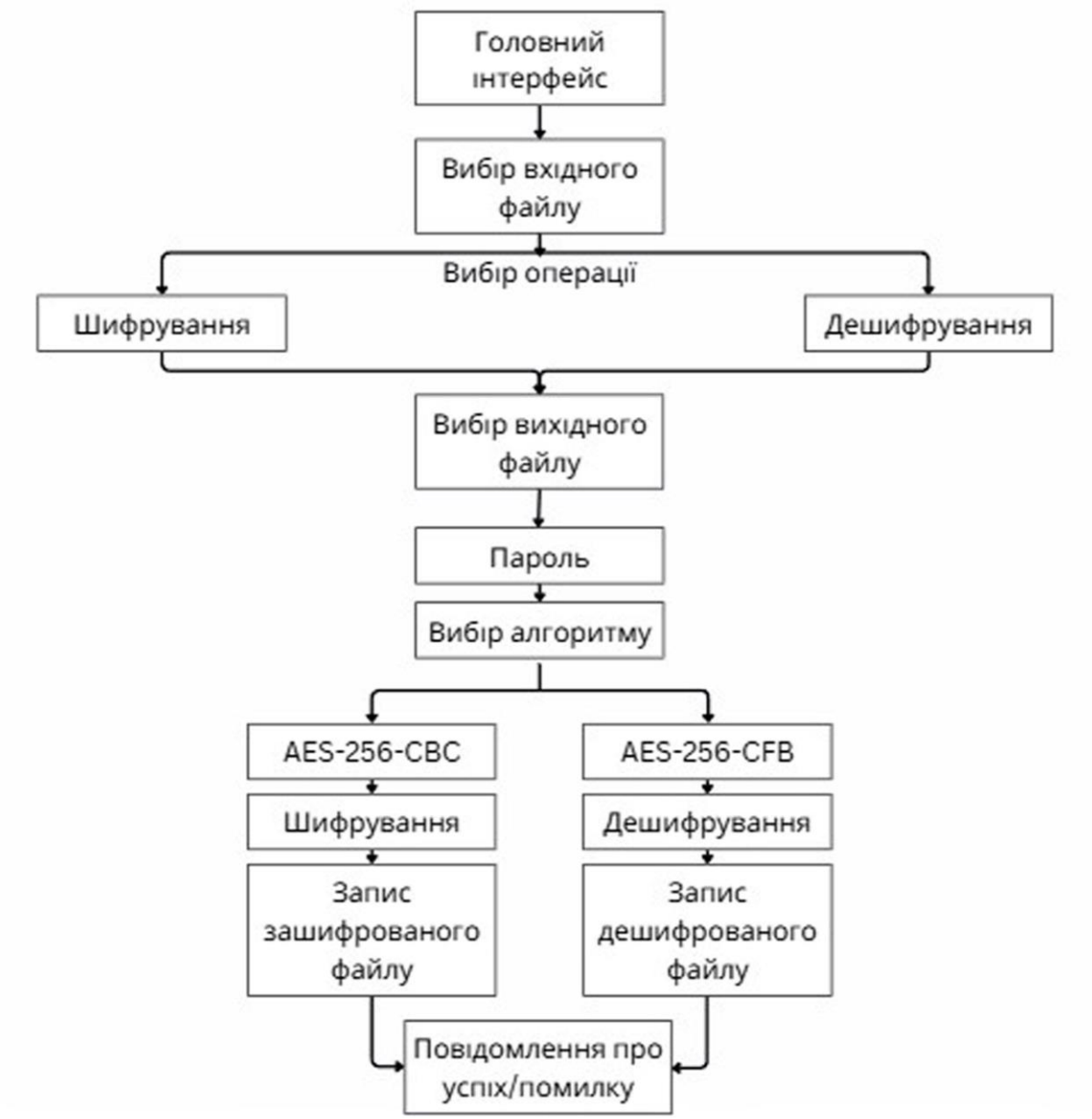
39. Що таке ланцюжок блоків шифрування CBC. 2024. С. 1. URL: https://www.techtarget.com/searchsecurity/definition/cipher-block-chaining (дата звернення: 20.04.2025).

40. Зворотний зв'язок зашифрованого тексту CFB. 2024. С. 1. URL: https://www.techtarget.com/searchsecurity/definition/ciphertext-feedback (дата звернення: 20.04.2025).

ДОДАТОК А

Копія графічної частини





						КРБКБ. 2101110.21.01.03 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Система шифрування файлів з різним ступенем таємності Блок-схема взаємодії з інтерфейсом	Літера	Маса / Масштаб
Розроб.		Божок Д.В.					
Перевір.		Тітова В.Ю.				Аркуш	Аркушів
Н.Контр.							
Т.Контр.		Мостовий С.В.				ХНУ, КБ-21-1	
Затв.		Ковоц Ю.П.					

					КРБКБ. 2101110.21.01.03 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Система шифрування файлів з різним ступенем таємності Блок-схема шифрування файлу	Літера / Маса / Масштаб
Розроб.		Божок Д.В.				
Перевір.		Тітова В.Ю.				Аркуш / Аркушів
Н.Контр.						
Т.Контр.		Мостовий С.В.				ХНУ, КБ-21-1
Затв.		Ківоц Ю.П.				

ДОДАТОК Б

Лістинг коду

```
import os
import hashlib
import json
import base64
from tkinter import Tk, Label, Entry, Button, filedialog, messagebox
from tkinter import ttk  # Для Combobox (вибору алгоритму)
from cryptography.fernet import Fernet
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import padding
import logging  # Для аудиту дій

# --- Конфігурація ---
KEY_STORAGE_METHOD = "in_memory"
SALT_LENGTH = 16
KDF_ITERATIONS = 100000
SUPPORTED_ALGORITHMS = ["AES-256-CBC", "AES-256-CFB"]
DEFAULT_ALGORITHM = SUPPORTED_ALGORITHMS[0]
LOG_FILE = "encryption_audit.log"
CHUNK_SIZE = 64 * 1024  # Розмір чанка для обробки великих файлів (64KB)
global root, input_entry, output_entry, password_entry, algorithm_combo

# --- Стан аутентифікації ---
current_user_token = None
root = None  # Оголошуємо root на верхньому рівні як глобальну змінну

# --- Налаштування логування ---
logging.basicConfig(filename=LOG_FILE, level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

# --- Функції аутентифікації ---
def authenticate_user():
    """Вікно для запиту логіна та пароля (лише для прикладу)."""
    auth_window = Tk()
    auth_window.title("Аутентифікація")

    username_label = Label(auth_window, text="Логін:")
    username_label.grid(row=0, column=0, sticky='w', padx=5, pady=5)
    username_entry = Entry(auth_window)
    username_entry.grid(row=0, column=1, padx=5, pady=5)

    password_label = Label(auth_window, text="Пароль:")
    password_label.grid(row=1, column=0, sticky='w', padx=5, pady=5)
    password_entry = Entry(auth_window, show="*")
    password_entry.grid(row=1, column=1, padx=5, pady=5)
```

```
    def login():
        username = username_entry.get()
        password = password_entry.get()
        if username == "user" and password == "password":
            global current_user_token, root
            current_user_token = base64.urlsafe_b64encode(os.urandom(32)).decode('utf-8')
            messagebox.showinfo("Успіх", "Аутентифікація успішна!")
            logging.info(f"Користувач {username} успішно аутентифікований.")
            auth_window.destroy()
            create_main_window() # Викликаємо функцію для створення основного UI
        else:
            messagebox.showerror("Помилка", "Неправильний логін або пароль.")

    login_button = Button(auth_window, text="Увійти", command=login)
    login_button.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

    root = Tk() #
    root.withdraw()
    auth_window.mainloop()
    if root.winfo_exists(): # Перевіряємо, чи головне вікно ще існує після закриття auth_window
        root.mainloop() # Запускаємо головний цикл подій, якщо вікно існує

def create_main_window():
    """Створює та відображає основне вікно програми."""
    global root, input_entry, output_entry, password_entry, algorithm_combo
    root.deiconify() # Показуємо основне вікно

    # Вхідний файл
    input_label = Label(root, text="Вхідний файл:")
    input_label.grid(row=0, column=0, sticky='w', padx=5, pady=5)
    input_entry = Entry(root, width=50)
    input_entry.grid(row=0, column=1, padx=5, pady=5)
    input_button = Button(root, text="Обрати", command=select_input_file)
    input_button.grid(row=0, column=2, padx=5, pady=5)

    # Вихідний файл
    output_label = Label(root, text="Вихідний файл:")
    output_label.grid(row=1, column=0, sticky='w', padx=5, pady=5)
    output_entry = Entry(root, width=50)
    output_entry.grid(row=1, column=1, padx=5, pady=5)
    encrypt_output_button = Button(root, text="Обрати (Шифр.)", command=lambda:
select_output_file("encrypt"))
    encrypt_output_button.grid(row=1, column=2, padx=5, pady=5)
    decrypt_output_button = Button(root, text="Обрати (Дешифр.)", command=lambda:
select_output_file("decrypt"))
    decrypt_output_button.grid(row=2, column=2, padx=5, pady=5)

    # Пароль
    password_label = Label(root, text="Пароль:")
    password_label.grid(row=3, column=0, sticky='w', padx=5, pady=5)
    password_entry = Entry(root, width=50, show="*")
    password_entry.grid(row=3, column=1, padx=5, pady=5)
```

```
    # Вибір алгоритму
    algorithm_label = Label(root, text="Алгоритм:")
    algorithm_label.grid(row=4, column=0, sticky='w', padx=5, pady=5)
    algorithm_combo = ttk.Combobox(root, values=SUPPORTED_ALGORITHMS,
state="readonly")
    algorithm_combo.set(DEFAULT_ALGORITHM)
    algorithm_combo.grid(row=4, column=1, padx=5, pady=5)

    # Кнопки операцій
    encrypt_button = Button(root, text="Зашифрувати", command=perform_encryption)
    encrypt_button.grid(row=5, column=1, sticky='ew', padx=5, pady=5)
    decrypt_button = Button(root, text="Дешифрувати", command=perform_decryption)
    decrypt_button.grid(row=6, column=1, sticky='ew', padx=5, pady=5)

def verify_token():
    """Перевіряє наявність токена (лише для прикладу)."""
    return current_user_token is not None

def get_user_identifier():
    """Повертає ідентифікатор користувача на основі токена (лише для прикладу)."""
    if current_user_token:
        return "authenticated_user"
    return "guest"

# --- Функції керування ключами (концептуально) ---
user_keys = {}

def load_key(user: str, file_identifier: str):
    if KEY_STORAGE_METHOD == "os_keyring":
        try:
            import keyring
            key = keyring.get_password("file_encryption", f"{user}_{file_identifier}")
            return base64.urlsafe_b64decode(key.encode('utf-8')) if key else None
        except ImportError:
            messagebox.showerror("Помилка", "Бібліотека 'keyring' не встановлена.")
            return None
    elif KEY_STORAGE_METHOD == "hsm":
        messagebox.showinfo("Інформація", "Інтеграція з HSM не реалізована в цьому прикладі.")
        return None
    elif KEY_STORAGE_METHOD == "kms":
        messagebox.showinfo("Інформація", "Інтеграція з KMS не реалізована в цьому прикладі.")
        return None
    elif KEY_STORAGE_METHOD == "in_memory":
        return user_keys.get(f"{user}_{file_identifier}")
    return None

def save_key(user: str, file_identifier: str, key: bytes):
    if KEY_STORAGE_METHOD == "os_keyring":
        try:
            import keyring
            key_b64 = base64.urlsafe_b64encode(key).decode('utf-8')
```

```
            keyring.set_password("file_encryption", f"{user}_{file_identifier}", key_b64)
        except ImportError:
            messagebox.showerror("Помилка", "Бібліотека 'keyring' не встановлена.")
    elif KEY_STORAGE_METHOD == "hsm":
        messagebox.showinfo("Інформація", "Інтеграція з HSM не реалізована в цьому прикладі.")
    elif KEY_STORAGE_METHOD == "kms":
        messagebox.showinfo("Інформація", "Інтеграція з KMS не реалізована в цьому прикладі.")
    elif KEY_STORAGE_METHOD == "in_memory":
        user_keys[f"{user}_{file_identifier}"] = key

def generate_derived_key(password: str, salt: bytes) -> bytes:
    password_bytes = password.encode('utf-8')
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,  # 256 біт
        salt=salt,
        iterations=KDF_ITERATIONS,
        backend=default_backend()
    )
    return kdf.derive(password_bytes)

# --- Функції шифрування та дешифрування (з обробкою великих файлів) ---
def encrypt_file(input_file: str, output_file: str, password: str, algorithm_name: str):
    user = get_user_identifier()
    salt = os.urandom(SALT_LENGTH)
    key = generate_derived_key(password, salt)
    save_key(user, input_file, key)

    try:
        with open(input_file, 'rb') as infile, open(output_file, 'wb') as outfile:
            outfile.write(algorithm_name.encode('utf-8') + b'\n')
            outfile.write(salt)
            iv = os.urandom(16)
            outfile.write(iv)

            if algorithm_name == "AES-256-CBC":
                cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
                encryptor = cipher.encryptor()
                padder = padding.PKCS7(algorithms.AES.block_size).padder()
                while True:
                    chunk = infile.read(CHUNK_SIZE)
                    if not chunk:
                        padded_final = padder.finalize()
                        outfile.write(encryptor.update(padded_final) + encryptor.finalize())
                        break
                    padded_chunk = padder.update(chunk)
                    outfile.write(encryptor.update(padded_chunk))
            elif algorithm_name == "AES-256-CFB":
                cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
                encryptor = cipher.encryptor()
                while True:
```

```
                chunk = infile.read(CHUNK_SIZE)
                if not chunk:
                    outfile.write(encryptor.finalize())
                    break
                outfile.write(encryptor.update(chunk))
        else:
            raise ValueError(f"Непідтримуваний алгоритм: {algorithm_name}")

    logging.info(f"Користувач {user} зашифрував файл '{input_file}' у '{output_file}' з алгоритмом '{algorithm_name}'.")
    messagebox.showinfo("Успіх", f"Файл '{os.path.basename(input_file)}' успішно зашифровано в '{os.path.basename(output_file)}' з алгоритмом '{algorithm_name}'.")

  except Exception as e:
    logging.error(f"Помилка шифрування файлу '{input_file}': {e}")
    messagebox.showerror("Помилка шифрування", f"Сталася помилка під час шифрування: {e}")

def decrypt_file(input_file: str, output_file: str, password: str):
  user = get_user_identifier()
  try:
    with open(input_file, 'rb') as infile, open(output_file, 'wb') as outfile:
      algorithm_name_bytes = infile.readline().strip()
      algorithm_name = algorithm_name_bytes.decode('utf-8')
      salt = infile.read(SALT_LENGTH)
      iv = infile.read(16)
      key = generate_derived_key(password, salt)

      if algorithm_name == "AES-256-CBC":
        cipher = Cipher(algorithms.AES(key), modes.CBC(iv), backend=default_backend())
        decryptor = cipher.decryptor()
        unpadder = padding.PKCS7(algorithms.AES.block_size).unpadder()

        encrypted_data = infile.read() # Зчитуємо весь файл

        decrypted_chunk = decryptor.update(encrypted_data)
        try:
          unpadded_data = unpadder.update(decrypted_chunk) + unpadder.finalize()
          outfile.write(unpadded_data)
        except Exception as e:
          logging.error(f"Помилка unpadding файлу '{input_file}': {e}")
          messagebox.showerror("Помилка дешифрування", f"Помилка unpadding: {e}. Можливо, неправильний пароль, пошкоджений файл або неправильний алгоритм.")
          return

      elif algorithm_name == "AES-256-CFB":
        cipher = Cipher(algorithms.AES(key), modes.CFB(iv), backend=default_backend())
        decryptor = cipher.decryptor()
        while True:
          chunk = infile.read(CHUNK_SIZE)
          if not chunk:
```

```
                    outfile.write(decryptor.finalize())
                    break
                outfile.write(decryptor.update(chunk))
            else:
                raise ValueError(f'Непідтримуваний алгоритм: {algorithm_name}')

        logging.info(f"Користувач {user} дешифрував файл '{input_file}' у '{output_file}'.")
        messagebox.showinfo("Успіх", f"Файл '{os.path.basename(input_file)}' успішно
дешифровано в '{os.path.basename(output_file)}'.")

    except Exception as e:
        logging.error(f"Помилка дешифрування файлу '{input_file}': {e}")
        messagebox.showerror("Помилка дешифрування", f"Помилка дешифрування: {e}.
Можливо, неправильний пароль, пошкоджений файл або неправильний алгоритм.")

# --- Графічний інтерфейс ---
def select_input_file():
    file_path = filedialog.askopenfilename()
    input_entry.delete(0, 'end')
    input_entry.insert(0, file_path)

def select_output_file(operation):
    initial_dir = os.path.dirname(input_entry.get())
    if operation == "encrypt":
        file_path = filedialog.asksaveasfilename(initialdir=initial_dir, defaultextension=".enc")
    else:
        base, ext = os.path.splitext(os.path.basename(input_entry.get()))
        suggested_name = os.path.join(initial_dir, base + "_decrypted" + ext)
        file_path = filedialog.asksaveasfilename(initialdir=initial_dir, initialfile=suggested_name)
    output_entry.delete(0, 'end')
    output_entry.insert(0, file_path)

def perform_encryption():
    print(f"Викликано perform_encryption. Токен: {current_user_token}")
    if not verify_token():
        messagebox.showerror("Помилка", "Будь ласка, спочатку пройдіть аутентифікацію.")
        return
    input_file = input_entry.get()
    output_file = output_entry.get()
    password = password_entry.get()
    selected_algorithm = algorithm_combo.get()
    if not input_file or not output_file or not password or not selected_algorithm:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля та оберіть алгоритм.")
        return
    encrypt_file(input_file, output_file, password, selected_algorithm)

def perform_decryption():
    print(f"Викликано perform_decryption. Токен: {current_user_token}")
    if not verify_token():
        messagebox.showerror("Помилка", "Будь ласка, спочатку пройдіть аутентифікацію.")
        return
```

```
    input_file = input_entry.get()
    output_file = output_entry.get()
    password = password_entry.get()
    if not input_file or not output_file or not password:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
        return
    decrypt_file(input_file, output_file, password)

# --- Ініціалізація GUI ---
if __name__ == "__main__":

    root = Tk() #
    root.title("Система Шифрування Файлів (Розширена)")
    root.withdraw()

    # Запит аутентифікації перед показом основного вікна
    authenticate_user()

    # Основний інтерфейс (створюється після аутентифікації)
    if root:
        # Вхідний файл
        input_label = Label(root, text="Вхідний файл:")
        input_label.grid(row=0, column=0, sticky='w', padx=5, pady=5)
        input_entry = Entry(root, width=50)
        input_entry.grid(row=0, column=1, padx=5, pady=5)
        input_button = Button(root, text="Обрати", command=select_input_file)
        input_button.grid(row=0, column=2, padx=5, pady=5)

        # Вихідний файл
        output_label = Label(root, text="Вихідний файл:")
        output_label.grid(row=1, column=0, sticky='w', padx=5, pady=5)
        output_entry = Entry(root, width=50)
        output_entry.grid(row=1, column=1, padx=5, pady=5)
        encrypt_output_button = Button(root, text="Обрати (Шифр.)", command=lambda:
select_output_file("encrypt"))
        encrypt_output_button.grid(row=1, column=2, padx=5, pady=5)
        decrypt_output_button = Button(root, text="Обрати (Дешифр.)", command=lambda:
select_output_file("decrypt"))
        decrypt_output_button.grid(row=2, column=2, padx=5, pady=5)

        # Пароль
        password_label = Label(root, text="Пароль:")
        password_label.grid(row=3, column=0, sticky='w', padx=5, pady=5)
        password_entry = Entry(root, width=50, show="*")
        password_entry.grid(row=3, column=1, padx=5, pady=5)

        # Вибір алгоритму
        algorithm_label = Label(root, text="Алгоритм:")
        algorithm_label.grid(row=4, column=0, sticky='w', padx=5, pady=5)
        algorithm_combo = ttk.Combobox(root, values=SUPPORTED_ALGORITHMS,
state="readonly")
```

```
algorithm_combo.set(DEFAULT_ALGORITHM)


algorithm_combo.grid(row=4, column=1, padx=5, pady=5)

# Кнопки операцій
encrypt_button = Button(root, text="Зашифрувати", command=perform_encryption)
encrypt_button.grid(row=5, column=1, sticky='ew', padx=5, pady=5)
decrypt_button = Button(root, text="Дешифрувати", command=perform_decryption)
decrypt_button.grid(row=6, column=1, sticky='ew', padx=5, pady=5)

root.mainloop()
```

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 5%**

	Document		Sum coincidence on the DB	
ID: 242999 Title: Система шифрування файлів з різним ступенем таємності Added in a DB: 2025-06-02 Authors: Божок Дмитро Валерійович Heads: Тітова В.Ю. Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	66379	1045	994 (1%)	14 (1%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Божок Дмитро Валерійович

Співавтор:

Назва: Система шифрування файлів з різним ступенем таємності

Науковий керівник:

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:6.7%

Коефіцієнт подібності 2:0.3%

Мікропробіли: 0

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-03 04:13:23.0

Після аналізу Звіту подібності констатую наступне:

☐ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

03.06.2025р.

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.

Божка Дмитра Валерійовича
ПІБ здобувача вищої освіти

Студента ФІТ, 4 курсу, групи КБ-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 31.08.2023, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.06.25
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система шифрування файлів з різним ступенем таємності

Автор: Божок Дмитро Валерійович

Спеціальність: 125 – Кібербезпека

Освітня програма: освітньо-професійна

Науковий керівник: Тітова Віра Юріївна, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою виявлення і запобігання плагіату StrikePlagiarism складає 93,3%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з Положенням про систему забезпечення академічної доброчесності у ХНУ (https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-systemu-zabezpechennya-akademichnoyi-dobrochesnosti.pdf, Додаток В) кваліфікаційна робота, виконана за освітньо-професійною програмою, кількісні показники рівня унікальності тексту у відсотках до загального обсягу матеріалу в якій складає 75-100 %, визнається роботою з високою унікальністю тексту: «Текст вважається унікальним і не потребує додаткових дій щодо запобігання неправомірним запозиченням».

Керівник роботи _____ Віра ТІТОВА

Гарант ОПП _____ Віктор ЧЕШУН

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

6. Оцінка графічного оформлення та пояснювальної записки роботи. Графічне оформлення виконане відповідно до теми кваліфікаційної роботи із дотриманням усіх стандартів. У загальному графічне оформлення виконане на достатньому технічному рівні. Пояснювальна записка відповідає нормам для її оформлення та вимогам

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. У пояснювальній записці багато наглядних пояснень. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої задачі.

8. Інші зауваження -

9. Оцінка дипломної роботи Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що робота заслуговує оцінки «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) __ завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки, д.т.н., професор Мартинюк Валерій Володимирович

« 06 » червня 2025. _____ (підпис)