

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для
шифрування користувацьких даних у хмарних сервісах”

Виконав здобувач вищої освіти
IV курсу, групи КБ-23МБ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Васи́леґа Д.Р.
«___» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Улічев О.С.
«___» _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 125 "Кібербезпека"
Освітньо-професійна (освітньо-наукова) програма "Кібербезпека"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Німенку Дмитру Олеговичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах*

2. Керівник роботи *Улічев Олександр Сергійович, канд. техн. наук, доцент*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 127-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту *22.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи кібербезпеки *1 аркуш*

Функціональна схема системи кібербезпеки *1 аркуш*

Діаграма процесів *1 аркуш*

Блок-схема алгоритму роботи додатку *3 аркуша*

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	22.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Улічев О.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Василега Д.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Василега Д.Р. Програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для шифрування користувацьких даних у хмарних сервісах.

Метою розробки є програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах.

Результат роботи – програмна реалізація системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual Studio, мовою JavaScript.

Ключові слова: кібербезпека, хмара

ABSTRACT

Vasylieha D.R. Cybersecurity system software for encrypting user data in cloud services. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this bachelor's qualification, software has been developed that is intended for encrypting user data in cloud services.

The purpose of the development is to develop a cybersecurity system software for encrypting user data in cloud services.

The result of the work is a software implementation of a cybersecurity system for encrypting user data in cloud services.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows 10/11.

The program is developed in the environment of Visual Studio, in JavaScript language.

Keywords: cybersecurity, Cloud

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.....	9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	21
2.3 Розгорнута постановка завдання	22
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	25
3.1 Опис функціонування системи	25
3.2 Розробка структурної схеми.....	27
3.3 Розробка функціональної схеми	31
3.4 Розробка діаграми процесів.....	34
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	37
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	37
4.2 Захист розробленого програмного забезпечення.....	47
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	50
6 ОСНОВНІ ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61

					ВКРБ-125.26.0033.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата	Програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах	Літ.	Аркуш	Аркушів
Розроб.	Василега Д.Р.					Б	1	65
Перев.	Улічев О.С.							
Н.контр.	Коваленко А.С.					ЦНТУ КБ-23МБ		
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПК – Персональний комп'ютер;
ПЗ – Програмне забезпечення;
AEAD – Authenticated Encryption with Associated Data;
AES – Advanced Encryption Standard;
AI/ML – Artificial Intelligence/Machine Learning;
Amazon EC2 – Elastic Compute Cloud;
Amazon S3 – Simple Storage Service;
API – Application Programming Interface;
AVI – Audio Video Interleave;
AWS – Amazon Web Services;
Azure VM – Virtual Machine;
CMEK – Customer-Managed Encryption Keys;
CSPM – Cloud Security Posture Management;
DOCX – Office Open XML Document;
DoS/DDoS – Denial of Service/Distributed Denial of Service;
EFS – Elastic File System;
GCM – Galois/Counter Mode;
GDPR – General Data Protection Regulation;
GIF – Graphics Interchange Format;
HIPAA – Health Insurance Portability and Accountability Act;
HTML – HyperText Markup Language;
HTTP/HTTPS – HyperText Transfer Protocol /HyperText Transfer Protocol Secure;
IaaS – Infrastructure-as-a-Service;
IAM – Identity and Access Management;
IDS/IPS – Intrusion Detection and Prevention Systems;

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

ISO – International Organization for Standardization;
 IV – Initialization Vector;
 JPG / JPEG – Joint Photographic Experts Group;
 JWT – JSON Web;
 KMS – Key Management Service;
 MFA – Multi-Factor Authentication;
 MP3 – MPEG-1 Audio Layer 3;
 MP4 – MPEG-4 Part 14;
 NAS – Network Attached Storage;
 NFS – Network File System;
 PaaS – Platform-as-a-Service;
 PBKDF2 – Password-Based Key Derivation Function 2;
 PDF – Portable Document Format;
 PNG – Portable Network Graphics;
 SaaS – Software-as-a-Service;
 SDK – Software Development Kit;
 SMB – Server Message Block;
 TLS – Transport Layer Security;
 TXT – Text File;
 vNET – Virtual Network;
 WAF – Web Application Firewall;
 WAN – Wide Area Network;
 WAV – Waveform Audio File Format;
 IT – Information Technology;
 .Enc – Encryption;
 .EXE – executable.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

ВСТУП

Актуальність теми. У сучасних умовах інформаційних технологій хмарні сервіси стали невід'ємною частиною цифрової інфраструктури. Вони забезпечують зберігання, обробку та передачу значних обсягів користувацьких даних, що зумовлює підвищені вимоги до їх захисту.

Особливістю хмарних сервісів є те, що дані користувачів зберігаються та обробляються на віддалених серверах, які можуть фізично знаходитися у різних географічних локаціях. Це створює додаткові ризики, пов'язані з передачею даних через мережу Інтернет, спільним використанням ресурсів та залежністю від постачальників хмарних послуг.

У 2026 році все більше компаній переходять на хмарні сервіси – останні дослідження показують, що 54% усіх даних у хмарі є чутливими, проте лише 8% організацій шифрують понад 80% своїх даних. Це означає, що переважна більшість компаній залишають критично важливу інформацію вразливою до атак, витоків та внутрішніх інцидентів. [1]

Мета й завдання дослідження. Метою даної дипломної роботи є розробка програмного забезпечення для безпечного зберігання файлів у хмарному сховищі із використанням сучасних криптографічних методів. Представлена робота являє собою опис проєктування додатку з описом всіх його можливостей. У роботі використано сучасні технології веб-розробки, зокрема клієнтські засоби обробки даних, серверні рішення, алгоритм шифрування та хмарний сервіс, що дозволяє створити ефективну та безпечну систему зберігання інформації.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми забезпечують ефективне розв'язання завдань, пов'язаних із реалізацією систем шифрування користувацьких даних у хмарних середовищах.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Кібербезпека є однією з ключових складових функціонування сучасних інформаційних систем та користувацьких сервісів. У зв'язку зі стрімким розвитком цифрових технологій, зростанням обсягів передавання та зберігання даних, а також поширенням онлайн-сервісів, питання захисту інформації набуває особливої актуальності.

Під цим терміном розуміють сукупність методів, засобів і технологій, спрямованих на забезпечення захисту інформації, інформаційних систем та мереж від несанкціонованого доступу, пошкодження або знищення. Основною її метою є гарантування трьох базових властивостей інформації:

Конфіденційність – захист інформації від несанкціонованого доступу. Це означає, що лише авторизовані користувачі можуть отримати доступ до даних та використовувати їх.

Цілісність – забезпечення точності та повноти інформації. Це передбачає захист даних від несанкціонованих модифікацій, видалення чи знищення.

Доступність – гарантування безперебійного доступу до інформації та систем авторизованим користувачам. Тобто системи мають бути стійкими до збоїв та атак. [2]

Хмара та кібербезпека тісно пов'язані поняття, оскільки хмарні обчислення часто передбачають зберігання та доступ до даних і додатків через інтернет, що може створювати нові ризики та виклики кібербезпеки. Хмара та кібербезпека потребують уважної уваги для забезпечення безпеки даних і додатків у хмарі. [30]

У світі, де дані стали новою валютою, хмарна безпека – це не просто технічний інструмент, а основа довіри до цифрової інфраструктури. Хмарна

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

безпека являє собою комплексну багаторівневу систему, що поєднує в собі технологічні рішення, організаційні політики та превентивні практики, спрямовані на захист цілісності, доступності та конфіденційності інформаційних активів, розміщених у хмарі. Простими словами, це все, що допомагає зробити роботу з хмарними сервісами безпечною: від шифрування файлів – до контролю того, хто й коли має до них доступ.

Як ми вже зрозуміли, хмарні сервіси на сьогодні є однією з основних платформ для зберігання та обробки користувацьких даних. Їх широке використання зумовлене масштабованістю, доступністю та зручністю, однак разом із перевагами вони створюють нові виклики у сфері кібербезпеки.

Під викликами розуміється загрози, які спрямовані на несанкціоноване проникнення в інформаційну систему окремого користувача або корпоративну мережу. Зловмисники постійно вдосконалюють свої методи атак, щоб уникати виявлення й використовувати нові вразливості. Однак деякі з цих методів досить поширені, що дозволяє до них можна підготуватися. [3]

Наявність специфічних ризиків у хмарних середовищах зумовлює необхідність детального аналізу загроз та вразливостей, що можуть призвести до компрометації користувацьких даних. Саме розуміння цих загроз є основою для впровадження ефективних механізмів захисту, зокрема шифрування інформації. У зв'язку з цим давайте розглянемо основні загрози та вразливості, характерні для хмарних сервісів.

Криптоджекінг. Криптоджекінг – це досить новий тип кіберзагроз для хмар. Зловмисники можуть отримувати доступ до хмарних систем і використовувати їхню обчислювальну потужність для майнінгу. Криптоджекінг виявити складно. Коли хакери використовують ресурси вашої хмарної системи, робота сервісів може сповільнюватися, і здається, що це відбувається через погане підключення до Інтернету або відсутність необхідних оновлень. Реальна ж причина проблеми стає зрозумілою пізніше, тому ігнорувати цей тип кіберзагрози для хмар точно не варто.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

Витоки даних. Можливо, найбільш поширеною кіберзагрозою для хмар є саме витік даних. Кіберзлочинці можуть отримати несанкціонований доступ до хмарної мережі або використати програми для перегляду, копіювання та передачі даних стороннім особам. Втрата інформації може призвести до порушення GDPR і великих штрафів для бізнесу, а водночас до репутаційного збитку та втрати довіри клієнтів. Саме тому одним із завдань має стати забезпечення безпеки хмарного сховища. [4]

Таким чином, хмарні сервіси, незважаючи на свої переваги, залишаються вразливими до широкого спектра кіберзагроз. Це обумовлює необхідність застосування надійних механізмів захисту, зокрема ефективних методів шифрування користувацьких даних як під час передавання, так і під час їх зберігання. Тому було вирішено розробити програмне забезпечення для шифрування користувацьких даних у хмарних сервісах.

Розроблене програмне забезпечення призначене для забезпечення кібербезпеки користувацьких даних шляхом їх шифрування, безпечного зберігання та передачі до хмарного сервісу. Основною метою системи є захист інформації від несанкціонованого доступу, витоку, модифікації або знищення, а також забезпечення конфіденційності, цілісності та доступності даних.

Система реалізує механізми криптографічного захисту, що включають шифрування файлів перед їх передачею у хмарне середовище, дешифрування при отриманні, а також перевірку цілісності даних за допомогою криптографічних хеш-функцій. Додатково забезпечується автентифікація користувачів і контроль доступу їх до ресурсів системи.

1.2 Область застосування

Область практичного застосування розробленого програмного забезпечення охоплює широкий спектр інформаційних систем, які використовують хмарні технології.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

Дана система може бути використана в діяльності пов'язаній з обробкою конфіденційної інформації. Використання системи дозволить підвищити рівень захисту даних, знизити ризики кіберзагроз та забезпечити відповідність сучасним вимогам інформаційної безпеки. Наприклад, корпоративні інформаційні системи, системи електронного документообігу, персональні хмарні сховища.

Таким чином, розроблена система є універсальним інструментом для забезпечення безпечної роботи з даними у хмарних середовищах та може бути інтегрована в існуючі інформаційні інфраструктури різного масштабу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для відеонагляду мережі кафе з використанням камер Axis, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ-2026

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
						8
Вим.	Арк.	№ докум.	Підпис	Дата		

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Хмарне сховище – це модель зберігання даних, у якій цифрові дані розміщуються у віртуальних пулах ресурсів, що охоплюють кілька серверів, часто розташованих у різних місцях (датацентрах). Фізичну інфраструктуру зазвичай обслуговують власники – хостингові компанії, які забезпечують доступ до даних і їхню безпеку.

Послуги хмарного сховища можуть бути доступні через інтерфейс вебсервісу (API) або додатки, які використовують API. [6]

Хмарне сховище радикально змінило те, як організації обробляють свою інфраструктуру даних. Раніше компаніям доводилось витрачати величезні бюджети на фізичне обладнання та спеціальні ІТ-команди, тепер же можна отримати доступ до даних з будь-якої точки світу. Як свідчать дослідження аналітичного агентства Gartner, основним напрямком і двигуном розвитку ІТ-індустрії хмарні технології стали з кінця 2009 року. [12]

На відміну від класичної кібербезпеки, де компанія захищає власні сервери та мережу, в хмарі ресурси фізично належать провайдеру (AWS, Azure, Google Cloud тощо). Тому відповідальність за безпеку розподіляється між провайдером, який відповідає за безпеку самої інфраструктури та користувачем, який в свою чергу відповідає за налаштування доступу, зберігання даних і контроль над тим, хто може до них дістатися.

Хмарні сховища можна розділити на чотири категорії: публічне, приватне, гібридне та спільнотне. [8]

Концепція публічного хмарного сховища базується на використанні

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

спільної IT-інфраструктури, що перебуває під повним управлінням глобальних сервіс-провайдерів. Прикладами є такі компанії, як AWS, Google Cloud і Microsoft Azure, які керують величезними центрами обробки даних, де кілька організацій зберігають свої дані поруч.

У цьому варіанті високий рівень доступності та відмовостійкості досягається шляхом механізму надмірності: дані автоматично реплікуються між різними вузлами та серверами. Це гарантує безперебійну роботу системи навіть у разі критичних збоїв апаратної частини або проведення технічних робіт.

Користувачі взаємодіють з хмарним середовищем на рівні абстракції – через API, спеціалізовані веб-консолі або інтегровані програмні рішення. Прямий контроль на фізичному рівні залишається повністю на провайдері.

Ця модель є найбільш вигідною з точки зору фінансів. Серед ключових переваг варто виділити:

По-перше вам не потрібно інвестувати у власне обладнання чи наймати штат технічних фахівців для його обслуговування.

По-друге завдяки величезним потужностям провайдера, вартість зберігання одиниці даних залишається мінімальною.

Загальнодоступна хмара ідеально підходить для компаній, яким потрібне гнучке та масштабоване рішення для зберігання даних.

Приватна хмара – це модель, за якої організація отримує у власне розпорядження виділену інфраструктуру. Вона може базуватися як на власних потужностях компанії, так і на ізольованих серверах стороннього провайдера. Головна особливість такого підходу в тому, що дані знаходяться в межах корпоративного інтранету, який в свою чергу захищається внутрішніми брандмауерами.

Для підвищення гнучкості активно впроваджується віртуалізація. Завдяки ній можна створити масштабовані середовища з доступом тільки через захищені приватні канали.

Така модель використовується в сферах, де досить сурові стандарти

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

безпеки. Наприклад медицина, де суворе дотримання протоколів, таких як НІРАА є обов'язковим. Також – державний сектор, де вимоги до конфіденційності найвищі.

Проте, у цієї моделі також є недоліки. Основними є:

1. Фінансове навантаження організації, яка може сплачувати всю виділену потужність, а використовувати лише частину.

2. Керування такою системою вимагає високої кваліфікації технічного персоналу.

Далі на черзі гібридне хмарне сховище. Воно базується на поєднанні приватних та загальнодоступних можливостей, що дозволяє гнучко модифікувати ресурси залежно від їх пріоритету. Щоб це забезпечити у цій архітектурі зазвичай корпоративні додатки та бази даних зазвичай ізолюються у приватній хмарі, а для обробки великої і менш важливої кількості даних використовуються публічні провайдери.

Давайте наведемо приклади такої гнучкості даної моделі:

1. Забезпечення оптимізації зберігання, наприклад, фінансові звіти та персональні дані клієнтів залишаються під повним контролем організації, а маркетинговий контент чи архіви переносяться у публічне сховище для зниження витрат.

2. Така модель є стандартом побудови для систем аварійного відновлення. В такому випадку хмара буде використана віддалене місце для реплікації та резервного копіювання. [8]

Проте використання такої моделі також стикається з викликами. Зокрема виникає потреба у окремих допоміжні інструменти для обробки переміщення даних, їх синхронізації. Також політики безпеки мають бути уніфікованими між різними середовищами. Це вимагає від штату фахівців відповідних знань в обох типах платформ.

Останнім типом є хмарне сховище спільноти, воно представляє собою спеціалізовану модель розгортання орієнтована на групу організацій із подібними

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

обчислювальними потребами, галузевими стандартами або вимогами до безпеки. Така структура створює високо захищене середовище для безпечного обміну даними між учасниками.

Модель ідеально підходить для сфер із жорстким державним регулюванням, як-от фінансовий сектор, охорона здоров'я або державні установи. В принципі, ця модель і використовується в тих випадках, коли не вистачає стандартів захисту публічних хмар

Основна перевага спільотної хмари полягає у розподілі витрат на підтримку високих стандартів. Кілька суб'єктів, стикаючись з ідентичними викликами у сфері кібербезпеки, спільно інвестують у розробку архітектури, що задовольняє їхні специфічні потреби. Таким чином ми отримуємо рівень захисту, близький до приватного сховища, за значно менші кошти завдяки колективній оренді та обслуговуванню ресурсів.

Тепер давайте перейдемо до розгляду технологій, які організують наші дані у хмарах середовищах.

Те, як ваші дані організовані та доступні, впливає на продуктивність, вартість і сумісність із вашими програмами. Система зберігання служить базовою інфраструктурою для керування та організації даних у хмарних середовищах, забезпечуючи ефективну обробку даних і доступність. Три основні типи організації даних: зберігання об'єктів, файлове сховище, блокове сховище.

Зберігання об'єктів служить для роботи з величезними масивами неструктурованих даних. Замість звичних ієрархій, тут кожен елемент зберігається, як окремий об'єкт у адресному просторі.

Особливістю такого типу є те, що замість шляху до файлу використовується унікальний ідентифікатор. Кожен об'єкт може містити розширені метадані, що описують його вміст(автор, рівень доступу, термін зберігання).

Основні приклади сервісів, що використовують таку модель: Amazon S3, Google Cloud Storage, Azure Blob Storage.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

Цей тип ідеально підходить для архівації, зберігання медіафайлів та великих наборів даних.

Але такий тип також має свої недоліки. Він має більшу затримку, і не підтримує операції на рівні файлів. Ця модель не придатна для баз даних або систем, що потребують миттєвого оновлення інформації.

Наступним є файлове сховище. Ця модель найбільш звична для користувача. Зберігання файлів організовує дані в ієрархічній структурі папок, які більшість користувачів розпізнають зі своїх настільних комп'ютерів.

Доступ до даних здійснюється через стандартні мережеві протоколи, такі як NFS (для Unix-систем) або SMB (для Windows).

Часто реалізовується у вигляді мережевих сховищ. Мережеве сховище (NAS) – це поширене рішення для зберігання файлів, яке забезпечує ієрархічну організацію папок і файлів за допомогою цих протоколів.

Серед відомих хмарних рішень можна виділити: Amazon EFS, Azure Files і Google Cloud Filestore. Такий мет зберігання забезпечує одночасний доступ до спільних ресурсів для багатьох користувачів. Це оптимальний варіант для корпоративного документообігу та спільної розробки ПЗ.

Це робить зберігання файлів ідеальним для спільної роботи, спільних програм і сценаріїв, де вам потрібна традиційна поведінка файлової системи в хмарі.

Останнім на черзі є блокове сховище. Блокове сховище розділяє дані на блоки фіксованого розміру, кожен з яких має унікальну адресу, і представляє їх серверам як необроблені томи сховища.

Блокова модель є найбільш низькорівневою та продуктивною. Ваша операційна система або програми бачать блокове сховище як звичайний «жорсткий диск», який вона може формувати під будь-яку файлову систему та використовувати для баз даних, програм або завантажувальних томів.

Це ідеальний варіант, якщо для програм потрібен швидкий та послідовний доступу до даних. Використання технологій NVMe дозволяє мінімізувати

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		13

затримки та забезпечити стабільно високу швидкість обміну даними.

Це доволі гарне рішення для систем баз даних, системи планування ресурсів підприємства, через низьку щатримку та високу пропускну здатність.

Приклади включають Amazon EBS, Azure Disk Storage і Google Persistent Disk.

Блокове сховище дає вам найбільший контроль над тим, як дані організовані та доступні. Багато корпоративних додатків призначені для роботи з блоковим сховищем.

Перейдемо до огляду того, яким чином забезпечується безпека в хмарних середовищах.

Рівень захищеності інформаційного середовища безпосередньо залежить від обраної моделі хмарного сервісу, оскільки кожна з них передбачає специфічний розподіл зон відповідальності між провайдером послуг та кінцевим користувачем. [5]

Відповідно до загальноприйнятої класифікації, виділяють три основні рівні абстракції хмарних обчислень:

1. SaaS (Software-as-a-Service). У даній моделі провайдер забезпечує повний стек інфраструктури, включаючи програмні застосунки. Відповідальність клієнта обмежена рівнем управління обліковими записами, конфігурацією політик доступу та безпекою кінцевих пристроїв. Типовими представниками є екосистеми Google Workspace та Microsoft 365.

2. PaaS (Platform-as-a-Service). Хмарна платформа надає середовище для розробки та виконання програмного забезпечення (ОС, бази даних, бібліотеки). Клієнт бере на себе повну відповідальність за безпеку прикладного коду, цілісність користувацьких даних та логіку доступу до сервісів (наприклад, AWS Elastic Beanstalk або Azure App Service).

3. IaaS (Infrastructure-as-a-Service). Найбільш гнучка модель, що надає низькорівневі ресурси (віртуалізація, сервери, мережа). Користувач несе максимальну відповідальність, що включає адміністрування операційних систем,

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

налаштування мережевих екранів (Firewalls) та імплементацію методів шифрування. Прикладами є Amazon EC2 та Google Compute Engine. [5]

Ефективна хмарна безпека складається з кількох ключових компонентів. Кожен із них дуже важливий, але саме в комбінації вони надають той рівень безпеки, якого очікують:

1. Криптографічні методи захисту. Виступають превентивним засобом забезпечення конфіденційності. Належна імплементація шифрування гарантує неможливість дешифрування інформації сторонніми особами навіть при фізичному доступі до носіїв. Сучасним стандартом є захист даних у стані спокою та під час транспортування із використанням алгоритму AES-256.

2. Управління ідентифікацією та доступом (IAM). Централізовані системи IAM дозволяють реалізувати гранулярний контроль доступу на основі рольових моделей. Фундаментальним тут є принцип мінімальних привілеїв, який обмежує повноваження користувача виключно функціональною необхідністю.

3. Аудит та безперервний моніторинг. Системи логування забезпечують прозорість усіх операцій у хмарі, що є необхідним для ідентифікації аномалій та ретроспективного аналізу інцидентів. Це дозволяє підтримувати відповідність жорстким стандартам безпеки, таким як ISO 27001 або GDPR [5].

4. Інтелектуальні системи виявлення загроз. Використання рішень класу IDS/IPS у поєднанні з технологіями штучного інтелекту (AI) та машинного навчання (ML) дає змогу аналізувати мережевий трафік у реальному часі, виявляючи вектори атак (наприклад, Zero-day) та несанкціоновані зміни конфігурації інфраструктури.

5. Контроль конфігурацій. Оскільки більшість інцидентів у хмарі зумовлена помилками в налаштуваннях, критично важливим є регулярний аудит параметрів безпеки та своєчасне патч-менеджмент програмного забезпечення.

Криптографічне перетворення інформації вважається найбільш ефективним технічним методом уникнення ризиків у хмарі. Основна ідея полягає у трансляції вихідного тексту в нечитабельний шифротекст, розкодування якого

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		15

неможливе без наявності легітимного ключа.

Шифрування забезпечує:

1. Забезпечення конфіденційності. Криптографічне перетворення гарантує, що зміст інформації залишиться недоступним для сторонніх суб'єктів навіть у разі успішного фізичного або логічного проникнення до сховищ даних. Це є ключовим фактором захисту інтелектуальної власності, фінансової звітності та персональних відомостей користувачів.

2. Верифікація цілісності та автентичності. Окрім приховування змісту, сучасні криптосистеми реалізують режими AEAD (Authenticated Encryption with Associated Data), такі як AES-GCM. Це дозволяє не лише шифрувати дані, а й виявляти будь-які спроби їх несанкціонованої модифікації, гарантуючи, що отримувач оперує оригінальною та цілісною інформацією.

3. Відповідність нормативно-правовим стандартам. Використання стійкого шифрування є обов'язковою умовою для дотримання вимог національного законодавства (зокрема Закону України «Про захист персональних даних») та міжнародних регламентів, які визначають криптографію як пріоритетний захід безпеки для чутливої інформації.

4. Ешелонований захист на всіх етапах життєвого циклу даних::

– дані при зберіганні (Data at Rest). Шифрування даних, що зберігаються на дисках, у базах даних, об'єктних сховищах, архівах;

– дані при передачі (Data in Transit). Захист потоків інформації, що циркулюють у відкритих мережах між клієнтськими терміналами та хмарною платформою, шляхом використання криптографічних протоколів рівня TLS/SSL.);

– дані під час обробки (Data in Use). Забезпечення безпеки даних безпосередньо в процесі їх використання в оперативній пам'яті. Реалізація цього рівня стає можливою завдяки концепції конфіденційних обчислень та технологіям апаратно-ізольованих анклавів;

Для забезпечення цілісності та конфіденційності інформаційних активів у сучасних хмарних архітектурах застосовується широкий стек криптографічних

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

бібліотек, стандартизованих протоколів та спеціалізованих програмних комплексів. Інтеграція таких рішень дозволяє побудувати ешелоновану систему захисту, яка охоплює як рівень мережевої передачі даних, так і рівень їх довгострокового зберігання в об'єктних або блокових сховищах.

Механізм криптографічного перетворення передбачає вибір алгоритму, який перетворює вихідні дані в зашифрований формат за допомогою ключа. Алгоритм шифрування та ключ працюють разом, щоб зробити дані нечитабельними для неавторизованих осіб. Цей процес включає кілька етапів, починаючи з вибору алгоритму та закінчуючи генерацією ключа шифрування. Деякі алгоритми надають статичні ключі, тоді як інші динамічно оновлюються в кожному сеансі.

У сучасній криптографії виділяють дві фундаментальні парадигми захисту інформації: симетричне та асиметричне шифрування (детальний порівняльний аналіз функціональних характеристик наведено у табл. 2.1).

Симетричне шифрування базується на використанні єдиного секретного ключа для процесів інкапсуляції та реверсивного відновлення даних. Ключовою перевагою даного підходу є висока обчислювальна ефективність та значна пропускна здатність, що робить його оптимальним вибором для обробки великих масивів інформації в хмарних сховищах. Проте суттєвим недоліком симетричних систем є необхідність забезпечення абсолютно безпечного каналу для дистрибуції ключа між суб'єктами обміну, що в умовах відкритих мереж стає критичною вразливістю.

Асиметричне шифрування (криптографія з відкритим ключем) функціонує на основі математично пов'язаної пари ключів: публічного (Public Key) та приватного (Private Key). Публічний компонент використовується для зашифрування повідомлення і може поширюватися вільно, тоді як розшифрування можливе лише за допомогою секретного приватного ключа, який зберігається виключно у власника. Такий механізм забезпечує значно вищий рівень безпеки та вирішує проблему передачі секретів, хоча й потребує суттєво

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

більших обчислювальних ресурсів порівняно із симетричними аналогами [9].

Таблиця 2.1 – Порівняння симетричного та асиметричного шифрування.

Характеристика	Симетричне шифрування	Асиметричне шифрування
Ефективність	Висока швидкість обробки; мінімальні затрати обчислювальних ресурсів. Оптимально для масивів даних.	Відбувається повільніше і вимагає більше обчислювальних ресурсів, особливо на етапі дешифрування. Підходить для невеликих обсягів даних і більш чутливих операцій.
Використання ключів	Використовується єдиний секретний ключ для обох сторін.	Розподіл на пару: відкритий (Public) та закритий (Private) ключі.
Передача ключів	Потребує захищеного каналу для обміну ключем. Проблема дис-трибуції ключів є критичною вразливістю.	Відкритий ключ передається вільно. Тільки приватний ключ потребує суворої ізоляції.
Приклади використання	Він використовується для шифрування файлів, баз даних, передачі даних через мережу, а також в інших ситуаціях, де потрібна швидка обробка та шифрування великих обсягів даних.	Для захисту обміну ключами, цифрових підписів, а також даних в електронній пошті, під час онлайн-транзакцій та в інших випадках, коли необхідна безпечна передача конфіденційної інформації. [9]
Безпека ключів	Головною загрозою є перехоплення ключа, оскільки знання його дає зловмиснику можливість розшифрувати всі дані.	Відкритий ключ доступний кожному, і навіть якщо його перехоплять, зловмисник не зможе розшифрувати дані без закритого ключа.
Алгоритми	Приклади: AES, Стандарт шифрування даних.	Приклади: RSA, криптографія на основі еліптичних кривих.

Продовження таблиці 2.1

Характеристика	Симетричне шифрування	Асиметричне шифрування
Візуальна концепція	Взаємодія на основі спільного секрету, про який сторони домовляються заздалегідь.	Використання математично пов'язаної пари ключів для розділення функцій зашифрування та розшифрування.
Комбінація методів	Використовується у поєднанні з асиметричним для шифрування великих обсягів даних після безпечного обміну ключами	Використовується для початкового безпечного обміну симетричними ключами шифрування, дозволяє застосовувати більш швидкі алгоритми.

Сучасний ринок хмарних обчислень пропонує різноманітні інтегровані механізми захисту об'єктних сховищ. Серед хмарних платформ, що надають вбудовані механізми шифрування, можна виділити:

1. Amazon Web Services (AWS). Платформа використовує багаторівневу модель безпеки, де базовим інструментом захисту Data in Transit є протоколи TLS, а для Data at Rest – галузевий стандарт AES-256. Центральним елементом архітектури є сервіс AWS KMS, який забезпечує автоматизацію життєвого циклу ключів. Особливу увагу варто приділити можливості використання виділених апаратних модулів CloudHSM (FIPS 140-2 Level 3), що дозволяє відокремити зберігання ключової інформації від загальної інфраструктури провайдера [17].

2. Google Cloud Platform (GCP). Характеризується підходом «безпека за замовчуванням», де всі дані автоматично шифруються перед записом на диск. Відмінною рисою GCP є гнучкість у виборі моделі управління ключами: від повністю автоматизованого сервісу Cloud KMS до стратегії CMEK (Customer-Managed Encryption Keys). Останній варіант надає користувачу повний суверенітет над криптографічними операціями, дозволяючи самостійно визначати

політики ротації та доступу [18].

3. Microsoft Azure.Пропонує інтегровані засоби захисту через механізм Azure Storage Encryption. Для централізованої оркестрації секретів та ключів використовується сервіс Azure Key Vault. Система глибоко інтегрована з моделлю доступу на основі ролей, що дозволяє диференціювати права доступу до даних та криптографічних ресурсів, мінімізуючи ризики внутрішніх загроз [19].

Вибір конкретного методу шифрування залежить від моделі загроз, вимог до безпеки, регуляторних норм, потреб у функціональності хмарних сервісів та готовності клієнта брати на себе відповідальність за управління ключами.

Попри високий рівень розвитку сучасних криптосистем, процес захисту інформації в хмарі супроводжується низкою критичних проблем, які можуть нівелювати ефективність розроблених засобів:

По-перше, проектування надійної криптосистеми вимагає високої кваліфікації. Некоректне обрання режимів роботи алгоритмів або помилки у конфігурації хмарного периметра безпеки створюють вразливості, що призводять до потенційної компрометації даних.

По-друге, навіть найстійкіші алгоритми стають неефективними у разі використання слабких автентифікаційних даних або порушення протоколів зберігання приватних ключів. Недотримання політик ІБ залишається одним із головних векторів атак на хмарні сервіси.

По-третє, безпека зашифрованої інформації безпосередньо залежить від цілісності ключів. Їх компрометація або несанкціонований доступ до ключових матеріалів фактично девальвує використання будь-якого стійкого алгоритму.

По-четверте, впровадження суцільного шифрування неминуче впливає на продуктивність системи (навантаження на процесор). Знаходження оптимального балансу між криптостійкістю та швидкодією є складною інженерною задачею.

По-п'яте, обмеженням є залежність від сторонніх хмарних провайдерів, які можуть контролювати інфраструктуру та частково процеси обробки даних. Це створює додаткові ризики для конфіденційності інформації.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

В ході виконання дипломного проєкту було розроблено веб додаток, для забезпечення безпечного зберігання користувачьких файлів. Основною функціональністю системи є шифрування файлів на стороні клієнта, їх передача до хмарного сховища та подальше безпечне завантаження і дешифрування.

Для реалізації системи було обрано мову програмування JavaScript.

Причинами вибору послужили:

1. Універсальність застосування та широка підтримка браузерами. Дозволяє виконувати криптографічні операції без встановлення додаткового програмного забезпечення.

2. Ефективна робота з мережею та файлами.

3. Наявність сучасних API, зокрема було використано Web Crypto API, для реалізації криптографії.

Для серверної частини було використано середовище виконання Node.js.
[14]

Причинами вибору послужили:

1. Зручна інтеграція з сервісами хмарних послуг.

2. Забезпечує асинхронну, неблокуючу модель вводу/виводу.

3. Велика екосистема npm-пакетів.

Для створення серверної логіки було використано фреймворк Express.js.

Причинами вибору послужили:

1. Мінімалістичність і простота.

2. Підтримка middleware, для авторизації, обробки файлів.

3. Висока популярність і стабільність. [16]

Для зберігання файлів було використано сервіс Amazon S3. Цей сервіс було обрано по наступним причинам:

1. Висока надійність і масштабованість.

2. Глобальна доступність.
3. Інтеграція через офіційний SDK.
4. Підтримка контролю доступу.

Для забезпечення конфіденційності даних користувача використано стандарт AES-GCM. [10]

Причинами вибору стали:

1. Сучасний стандарт симетричного шифрування.
2. Ефективність та швидкодія.
3. Підтримка в Web Crypto Api. [15]

Для генерації ключів використано алгоритм PBKDF2, що дозволяє отримувати криптографічно стійкі ключі з пароля користувача.

Інші інструменти та бібліотеки:

1. fs (вбудований модуль Node.js), для збереження користувачів у JSON-файлі.
2. bcrypt – алгоритм хешування паролів, який використовується для безпечного зберігання паролів. [24]
3. cors – забезпечує підтримку міждомених запитів (Cross-Origin Resource Sharing), що дозволяє клієнтській частині взаємодіяти із сервером.
4. multer – використовується для обробки завантаження файлів на сервер.
5. jsonwebtoken – реалізує механізм автентифікації за допомогою JSON Web Token. [25]
6. Visual Studio Code – як основне середовище розробки.

2.3 Розгорнута постановка завдання

У зв'язку з розглянутими проблемами хмарних сервісів вище й необхідністю забезпечення конфіденційності та цілісності користувацьких даних у сучасних умовах, було розроблено програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

Метою розробки програмного забезпечення є забезпечення безпеки користувацьких файлів, шляхом реалізації наступних функцій:

1. Шифрування файлів на стороні клієнта.
2. Передачі зашифрованих даних у хмарне сховище.
3. Забезпечення доступу до файлів лише авторизованим користувачам.
4. Реалізації механізму дешифрування при завантаженні.

Для досягнення мети потрібно було вирішити такі задачі:

1. Розробити механізм реєстрації та автентифікації користувачів. Перш за все це забезпечить додатковий захист, а також за допомогою цього механізму ми реалізуємо розмежовану систему зберігання користувацьких файлів, тобто кожен користувач має доступ тільки до своїх файлів. Тому система виконує наступне: перевіряє токен користувача при кожному запиті, реалізує перевірку відповідності користувача та запитуваного файлу, забороняє доступ до файлів іншим користувачам. У кожного користувача буде свій каталог у хмарі.

2. Реалізувати генерацію криптографічного ключа на основі пароля користувача. Цей механізм реалізує другий ступінь захисту ваших файлів, для кожного файлу буде свій ключ з паролем який ввів користувач. Тому, якщо навіть зломисники, дізнаються ваш логін та пароль, все одно не зможуть отримати доступ до файлів.

3. Шифрування файлів перед передачею. Тобто використання підходу Zero-knowledge, при якому сервер не має доступу до відкритого вмісту файлів. Шифрування було реалізовано за допомогою алгоритму AES-GCM, також було реалізовано генерування унікального вектору ініціалізації (IV) для кожного файлу, гарантуючи, що однаковий відкритий текст перетворюється на різний шифротекст.

4. Передача та збереження файлів у хмарному середовищі. Програмне забезпечення має забезпечити надійне зберігання в хмарному сервісі. Для цього треба забезпечити механізм завантаження файлів на сервер, формування

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

унікального ідентифікатора файлу, організувати структуру зберігання файлів за іменем користувачів.

5. Вивантаження файлів із хмарного сховища та їх дешифрування. Система повинна забезпечити можливість отримання назад файлів користувачем, для цього в додатку реалізовано дешифрування файлів.

6. Обробка помилок та виняткових ситуацій. Система повинна коректно і зрозуміло реагувати на помилки. Було реалізовано обробку помилок на сервері та клієнті, додано інформування людини про причини помилок.

Результатом виконання роботи є програмне забезпечення, яке забезпечує захищене зберігання файлів у хмарному середовищі, гарантує конфіденційність даних, реалізує сучасні методи криптографічного захисту, може бути використана як основа для побудови більш складних систем кібербезпеки захисту даних користувачьких даних у хмарі.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		24

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Особливістю архітектури програмного забезпечення, що базується на концепції Zero-Knowledge (нульової розголоски), є те, що серверна сторона оперує виключно зашифрованими даними. Це мінімізує або повністю нівелює ризик витоку конфіденційної інформації користувачів. Навіть у випадку компрометації серверної частини програми або отримання зловмисниками повного доступу до баз даних і систем збереження, зловмисник отримає лише набір шифротекстів, дешифрування яких без унікальних ключів клієнта є практично неможливим [27].

З метою ізоляції потоків даних, чіткого розмежування файлових об'єктів різних користувачів, а також забезпечення ергономічного процесу завантаження та отримання даних, до складу програмного забезпечення було інтегровано модуль авторизації та автентифікації. Цей блок виконує кілька критично важливих функцій:

1. Логічне структурування сховища. забезпечує персоналізоване зберігання файлів у хмарному просторі, де для кожного верифікованого облікового запису динамічно виділяється та закріплюється ізольований каталог.

2. Багаторівневий захист. додає обов'язковий етап суворої верифікації суб'єкта, що суттєво підвищує загальний рівень захищеності вебзастосунку від несанкціонованого доступу.

Розроблена система є повнофункціональним вебзастосунком для захищеного зберігання та менеджменту файлів у хмарному середовищі, ключовою особливістю якого є реалізація криптографічних перетворень безпосередньо на стороні клієнта. Фундаментальною метою проєкту є безальтернативне забезпечення конфіденційності та цілісності даних користувача

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

за суворим принципом Zero-Knowledge. За такої схеми серверний бекенд виконує роль виключно сліпого транзитного та зберігаючого вузла, який не володіє ключами дешифрування, не має технічної можливості аналізувати вміст файлів та не може скомпрометувати інформацію користувача за жодних умов.

Архітектура розробленого рішення побудована на класичній клієнт-серверній моделі, інтегрований із передовими сервісами хмарної інфраструктури Amazon Web Services (AWS). Як базовий рівень зберігання даних (Data Storage Layer) використано високодоступне об'єктне сховище Amazon S3, у бакетах якого розміщуються виключно попередньо зашифровані файлові об'єкти.

Для реалізації надійних криптографічних операцій у середовищі веббраузера без залучення сторонніх ресурсомістких бібліотек було використано стандартизований інтерфейс Web Crypto API. Криптографічний конвеєр на стороні клієнта складається з таких етапів:

1. Генерація ключів (Key Derivation). Використовується функція формування ключа на основі пароля PBKDF2. Вона перетворює майстер-пароль користувача на стійкий криптографічний ключ, ускладнюючи атаки методом грубої сили (brute-force) за рахунок ітеративного хешування з використанням унікальної солі (salt).

2. Симетричне шифрування. Безпосередній захист файлів перед надсиланням у хмару здійснюється за допомогою алгоритму AES-GCM (Advanced Encryption Standard у режимі Galois/Counter Mode). Вибір цього режиму зумовлений тим, що він забезпечує автентифіковане шифрування (AEAD), гарантуючи не лише абсолютну конфіденційність вмісту файлу, а й контроль його цілісності (захист від несанкціонованої модифікації під час транспортування або зберігання).

Безпека сесій та контроль доступу до API на рівні взаємодії «клієнт-сервер» реалізовані за допомогою компактних та безпечних токенів JWT (JSON Web Tokens). На стороні сервера для захисту облікових даних користувачів впроваджено механізм незворотного хешування паролів за допомогою

криптографічної функції bcrypt. Це гарантує, що навіть у разі повної компрометації реляційної бази даних користувачів, автентифікаційні маркери та реальні паролі залишаться захищеними від швидкого відновлення через словникові атаки або «rainbow tables». Такий комплексний підхід формує ешелоновану систему захисту інформаційного вебзастосунку.

Перейдемо до опису схем програмного додатку для наглядного зображення роботи системи.

3.2 Розробка структурної схеми

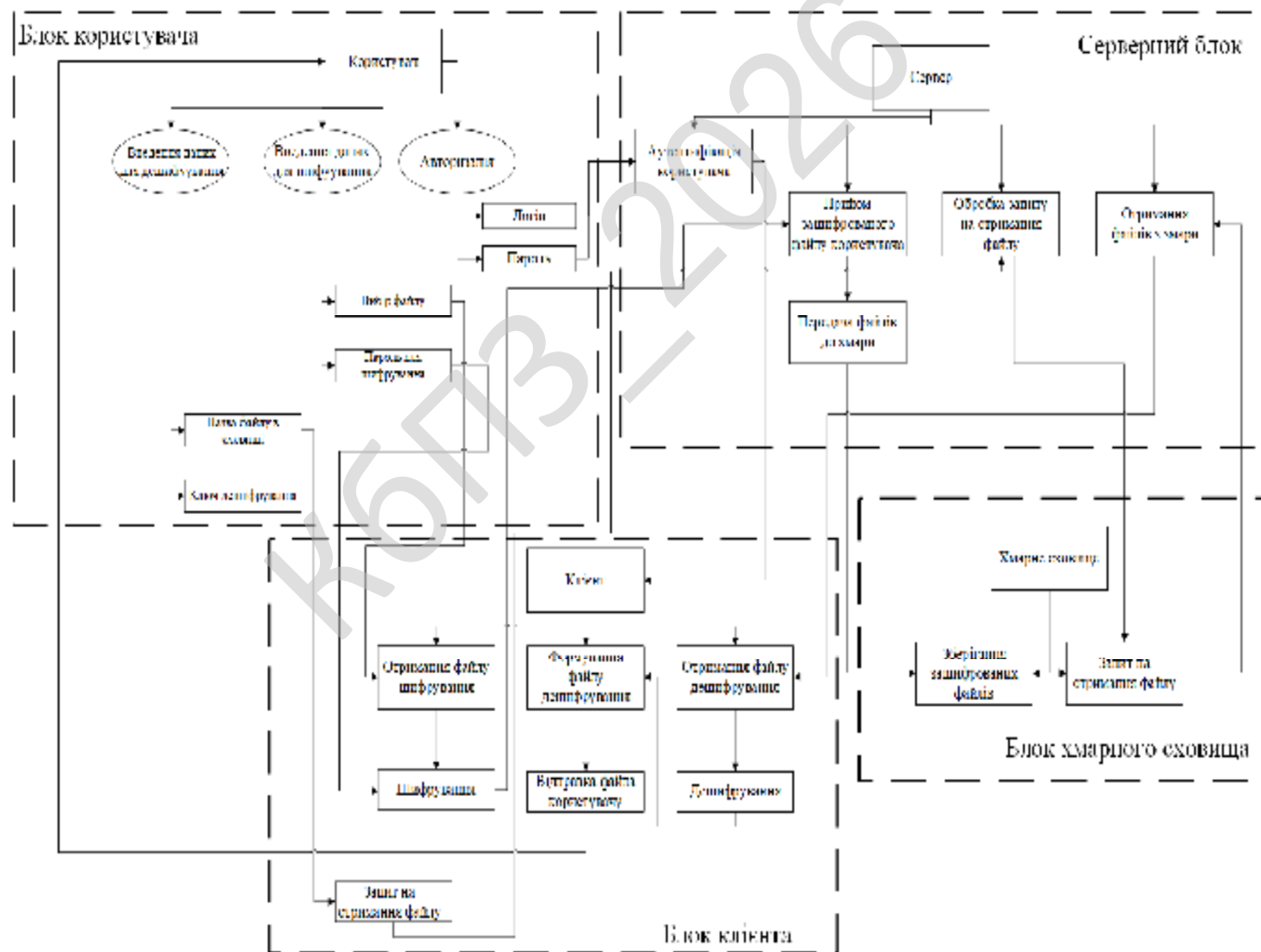


Рисунок 3.1 – Структурна схема системи

Зображена на рисунку 3.1 структурна схема побудована за принципом клієнт-серверної архітектури з використанням хмарного сховища. Компоненти системи складаються з таких блоків: користувач, клієнт, сервер та хмарне середовище.

Користувач взаємодіє з програмним комплексом реалізована через адаптивний веб-інтерфейс. Користувач має такі можливості:

1. Реєстрація та авторизація. Якщо користувач зайшов уперше, він проходить стандартну реєстрацію. Потрібно придумати унікальний логін та пароль для входу. Ці дані відправляються на сервер, де створюється обліковий запис. Логін користувача згодом використовується як назва його персональної папки в хмарі.

2. Завантаження файлів. Користувач натискає кнопку «Обрати файл» і вибирає будь-який документ чи медіафайл на своєму комп'ютері. Можна вибрати будь-який формат, який підтримує браузер, наприклад документи (PDF, DOCX, TXT), зображення (JPG, PNG, GIF), відео (MP4, AVI), аудіо (MP3, WAV) та архіви або exe-файли.

3. Введення пароля для шифрування. Окрім загального пароля для входу на сайт, користувач вводить окремий пароль для шифрування конкретного файлу. Цей пароль нікуди не відправляється через інтернет і не зберігається в жодній базі даних. Програма в браузері миттєво перетворює його на криптографічний ключ. На майбутнє користувач має запам'ятати пароль для свого файлу.

4. Завантаження файлу з хмари. Для завантаження зашифрованого файлу назад користувач вводить у відповідне поле назву свого файлу та формат в каталозі, де він збережений, також вводить свій пароль для файлу. Якщо дані вірні, то він отримує назад свій дешифрований файл.

Клієнтський блок працює на мові JavaScript у браузері користувача. На нього покладено найбільшу відповідальність за безпеку, оскільки тут реалізовано концепцію Client-Side Encryption (шифрування на стороні клієнта). Клієнтська частина реалізована у браузері та відповідає за такі функції:

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		28

1. Обробка введених даних. Сюди входить обробка подій користувача (натискання кнопок, вибір фалу, введення даних).

2. Виконання криптографічних операцій. Клієнтська частина виконує генерацію криптографічного ключа на основі введеного пароля користувача та шифрування та дешифрування.

При шифруванні браузер зчитує файл як сирий набір байтів (ArrayBuffer). Далі вбудований криптогенератор створює випадковий вектор ініціалізації (IV). Далі береться пароль користувача, додається випадкова salt, і за допомогою важкого алгоритму PBKDF2 (з великою кількістю ітерацій хешування) створюється міцний 256-бітний симетричний ключ. Наприкінці алгоритм AES-GCM (режим лічильника з автентифікацією Галуа) перетворює байти файлу на зашифрований файл і додає до неї спеціальну мітку цілісності (tag).

При дешифруванні браузер отримує з сервера зашифрований файл. Насамперед скрипт відділяє від нього вектор ініціалізації та мітку цілісності, які були прикріплені раніше. Далі з введеного паролю користувача, алгоритм PBKDF2 знову вираховує ключ, і запускається процес дешифрування через AES-GCM. Якщо мітка tag збігається, це означає, що файл ніхто не підмінив і не пошкодив у дорозі, і браузер збирає байти назад у готовий до використання файл.

3. Взаємодія з сервером. Цей підблок збирає зашифровані масиви даних, пакує їх у бінарний формат і відправляє на сервер через асинхронні HTTP-запити.

Серверна частина системи реалізована у вигляді веб-додатку, який виконує роль посередника між клієнтом та хмарним сховищем. Головна його особливість у тому, що він не має доступу до файлів користувача. Оскільки файли приходять до нього вже зашифрованими, сервер не має доступу до їхнього вмісту і займається виключно організаційними питаннями:

1. Автентифікація та авторизація користувачів. Коли користувач вводить логін та пароль для входу на сайт, сервер перевіряє їх через базу даних. Якщо все добре, він створює безпечний маркер доступу – jwt (JSON Web Token). Цей токен підписується секретним ключем сервера. Надалі клієнт прикріплює цей токен до

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

кожного свого запиту. Спеціальний фільтр на сервері (Middleware) автоматично відсікає будь-які запити, якщо у них немає правильного або валідного токена.

2. Передача файлів до хмарного середовища. Приймає зашифровані дані від користувача, перевіряє токен доступу, формує унікальний ключ з урахуванням логіну користувача, для подальшої назви каталогу в хмарі, передає файл у хмарне сховище.

3. Отримання файлів із хмарного середовища. Сервер перевіряє права доступу користувача, запитує файл із хмарного сховища, передає його клієнтській частині без змін для подальшого дешифрування.

Хмарне сховище використовується для фізичного зберігання файлів та забезпечення їх доступності. Сховище працює з серверною частиною програмного додатку. Функції хмарного сховища:

1. Зберігання зашифрованих файлів. У середині хмари простір розділений на логічні папки. Назва кожної папки – це логін конкретного користувача. Файли всередині цих папок лежать виключно в зашифрованому вигляді. Навіть якщо зловмисники якимось чином отримають прямий доступ до серверів хмарного провайдера, вони побачать лише випадковий набір бінарних даних, який неможливо прочитати без пароля шифрування, що зберігається в голові користувача.

2. Обробка запитів на отримання файлів. Хмара спілкується тільки з сервером за допомогою захищених API-ключів. Вона не приймає запити від звичайних користувачів напряму.

Головна перевага такої побудови системи полягає в тому, що ми повністю прибираємо слабку ланку у вигляді проміжних серверів. Зазвичай у стандартних хмарних сховищах файли шифруються вже на самому сервері, а отже, адміністратор сервера або хакер, який його зламав, мають доступ до ключів. У нашому випадку ключ генерується в оперативній пам'яті браузера користувача і там же знищується після закриття вкладки.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

3.3 Розробка функціональної схеми

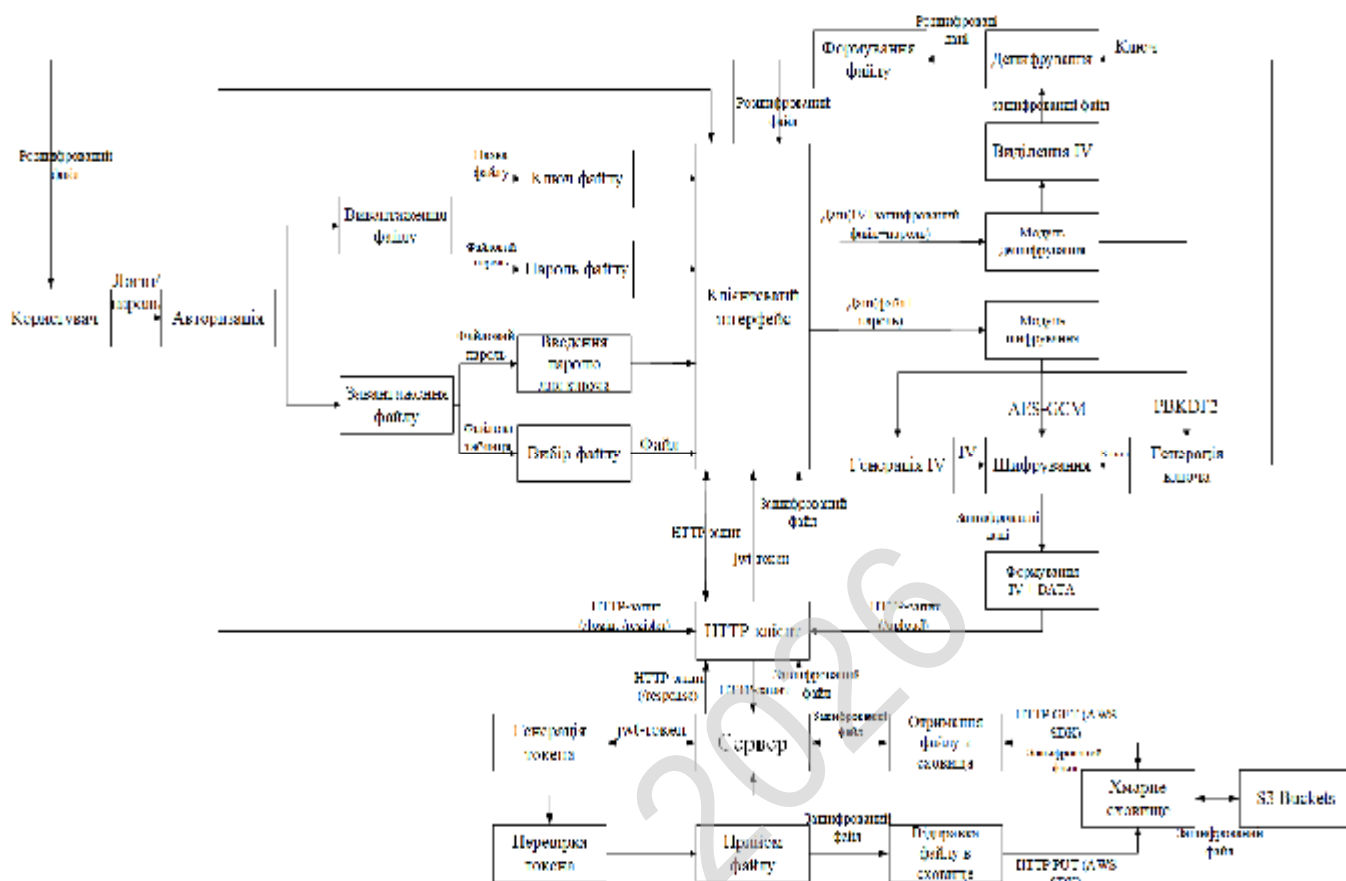


Рисунок 3.2 – Функціональна схема системи

Функціональна схема системи, що представлена на рисунку 3.2, детально демонструє архітектуру взаємодії між основними модулями, компонентами та сервісами програмного додатка. Кожен елемент схеми є автономним функціональним блоком, який відповідає за конкретний етап обробки даних, забезпечуючи загальну працездатність, безпеку та високу швидкість виконання операцій у системі.

Нижче наведено детальний аналіз та призначення кожного функціонального блока:

1. Користувач. Цей блок є початковою точкою взаємодії з системою. Він уособлює кінцевого користувача, який ініціює роботу з додатком. Доступ до основного функціоналу надається виключно після проходження внутрішньої

системи автентифікації та авторизації. Після успішного підтвердження особистих даних (логін та пароль), користувач отримує доступ до графічного інтерфейсу клієнтської частини додатка.

2. Введення даних. На даному етапі користувач вводить необхідні дані – обирає зашифрувати файл та відправити до хмари чи завантажити файл з хмари та дешифрувати його.

Для першої операції користувач обирає локальний файл на своєму ПК за допомогою файлового менеджера та задає пароль. Цей пароль є основою для майбутнього шифрування і не передається на сервер, тому користувач повинен надійно його запам'ятати.

Для завантаження зашифрованого файлу з хмари користувач ініціює запит на отримання файлу, вказуючи його унікальний ідентифікатор у хмарному сховищі (шлях до файлу, що складається з каталогу та назви), а також вводить пароль, який безпосередньо використовуватиметься для розшифрування.

3. Інтерфейс клієнту. Ключовий компонент архітектури, який функціонує на стороні користувача. Він виконує роль інтелектуального посередника між користувачем, локальними обчислювальними ресурсами та віддаленим сервером. Саме на стороні клієнтського інтерфейсу зосереджено весь комплекс криптографічних перетворень.

4. Модулі шифрування та дешифрування. Являє собою сукупність алгоритмів та методів захисту інформації.

Модуль шифрування приймає дані від користувача (файл та пароль) після чого переходить до генерації ключа. На основі введеного пароля виконується генерація криптографічного ключа. Ключ використовується для подальшого шифрування або дешифрування даних. Генерація вектора ініціалізації (IV) – робить кожне шифрування унікальним, навіть для одного і того ж файлу та використовується при подальшому дешифруванні. Шифрування виконано алгоритмом AES-GCM – симетричний блочний шифр AES з додаванням лічильника для забезпечення не тільки конфіденційності, а й цілісності. Після

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

чого формується файл для відправки на сервер через HTTP-запит.

Модуль дешифрування приймає від користувача пароль для файлу та зашифрований файл від серверу. Після чого відбувається генерація ключа, повторна генерація потрібна задля забезпечення архітектури при якій сервер не зберігає ключі користувачів. Після чого у блоці виділення IV відбувається перевірка вектору, для впевнення, що файл цілий. Далі відбувається процес дешифрування файлу після чого файл формується та користувач має змогу завантажити його собі на ПК.

5. HTTP-клієнт. Мережевий модуль, відповідальний за організацію стабільного та безпечного каналу зв'язку між клієнтським додатком та серверною частиною. Він здійснює валідацію, серіалізацію даних, формування заголовків (включаючи токени авторизації) та асинхронне відправлення HTTP-запитів, а також обробку HTTP-відповідей (статуси помилок, таймаути, тощо).

6. Сервер. Приймає запити від клієнту та виконує відповідні функції. Генерує токен користувача при авторизації та потім перевіряє його під час виконання операцій користувачем, що забезпечує конфіденційність. Якщо користувач шифрував файл, то після підтвердження валідності токenu приймає файл та відправляє до хмари через HTTP-запит PUT (AWS SDK). В іншому випадку, якщо користувач дешифрує файл, сервер надсилає запит HTTP-запит GET до хмари та отримує зашифрований файл, який передає клієнту.

7. Хмара. Кінцева точка збереження даних, реалізована на базі об'єктного сховища Amazon S3. Файл зберігається у зашифрованому вигляді, використовується унікальний ключ назви каталогу (логін користувача.). Хмара ізолювано обробляє команди від сервера на завантаження (PUT) або вивантаження (GET) об'єктів, забезпечуючи високу відмовостійкість та масштабованість системи зберігання.

Ми розглянули, як взаємодіють потоки даних при роботі функцій програмного забезпечення, перейдемо до розгляду схеми діаграми взаємодії процесів.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		33

3.4 Розробка діаграми процесів



Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма процесів, представлена на рисунку 3.3, детально відображає архітектуру взаємодії, послідовність виконання операцій та логіку роботи розробленого програмного забезпечення. Представлена схема візуалізує життєвий цикл сесії користувача, починаючи від моменту ініціалізації додатку і закінчуючи фінальним збереженням даних та завершенням роботи.

Вхідною точкою системи є процес «Початок», який відповідає за запуск клієнтського додатка та первинне налаштування середовища. Після цього відбувається перехід до стану «Вхід в систему». На даному етапі запускається підпроцес «Авторизація», який забезпечує перевірку облікових даних

користувача, підтвердження його прав доступу та забезпечення безпечного входу в систему.

Після успішного проходження авторизації користувач перенаправляється до «Інтерфейсу клієнта». Даний компонент є центральним ядром (вузлом) клієнтської частини додатка, який консолідує керування всіма подальшими операціями та забезпечує взаємодію з іншими підсистемами.

З метою забезпечення конфіденційності інформації, інтерфейс клієнта ініціює модуль «Шифрування», робота якого складається з наступних взаємопов'язаних технологічних етапів:

1. Генерація ключа. Створення унікального криптографічного ключа, що буде використаний для симетричного зашифрування інформації.

2. Генерація IV. Створення вектора ініціалізації, що гарантує унікальність шифротексту навіть при повторному шифруванні однакових блоків даних.

3. Формування IV+DATA. Фінальний етап шифрування, на якому згенерований вектор ініціалізації об'єднується із зашифрованими корисними даними в єдину структуру, готову для подальшої передачі або зберігання.

Зворотний процес отримання інформації реалізовано через модуль «Дешифрування», який виконує обробку отриманих масивів даних у такій послідовності:

1. Виділення IV. Із загального пакету даних вилучається вектор ініціалізації, необхідний для коректного запуску алгоритму зворотного перетворення.

2. Генерація (або відновлення) ключа. Отримання криптографічного ключа, відповідного до алгоритму шифрування.

3. Формування файлу. Безпосереднє розкодування інформації та відновлення початкового вигляду файлу для його подальшої демонстрації або збереження користувачем через інтерфейс клієнта.

Компонент «сервер» виступає в ролі ключового посередника (проміжного шару) між клієнтським інтерфейсом та хмарною інфраструктурою. До його

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		35

базових функцій належать:

1. Генерація токєну та перевірка токєну. Реалізація сесійного захисту та перевірка правомірності запитів від клієнта за допомогою механізму токєнів безпеки, що запобігає несанкціонованому доступу до даних.

2. Менеджмент файлових потоків. Координація процесів «Завантаження файлу» та «Вивантаження файлу» під час обміну даними з клієнтом та хмарою.

Компонент «Хмарне сховище» виступає кінцевим репозиторієм для надійного та ізольованого зберігання інформації. Сховище функціонує у двох режимах:

1. Прийом даних. Отримання зашифрованого файлового пакету від сервера та його довготривале розміщення.

2. Видача даних. Надсилання збереженого файлу назад на сервер за відповідним легітимним запитом від авторизованого користувача.

Після виконання всіх необхідних операцій (шифрування, передачі, дешифрування або збереження даних) та закриття поточної сесії користувачем, інтерфейс клієнта передає керування на фінальний стан «Кінець». Це сигналізує про успішне завершення життєвого циклу роботи додатку та коректне вивільнення задіяних системних ресурсів.

Таким чином, представлена діаграма взаємодії процесів дозволяє комплексно оцінити архітектурну логіку додатку та простежити шлях проходження даних між клієнтом, сервером і хмарним сховищем. Організація системи навколо центрального інтерфейсу клієнта забезпечує модульність архітектури, де криптографічні операції шифрування та дешифрування чітко ізольовані від процесів збереження та авторизації. Описана взаємодія підтверджує, що архітектурні рішення додатку орієнтовані на високий рівень безпеки. Перейдемо до опису алгоритмів та розглянемо блок-схеми роботи програмного забезпечення.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

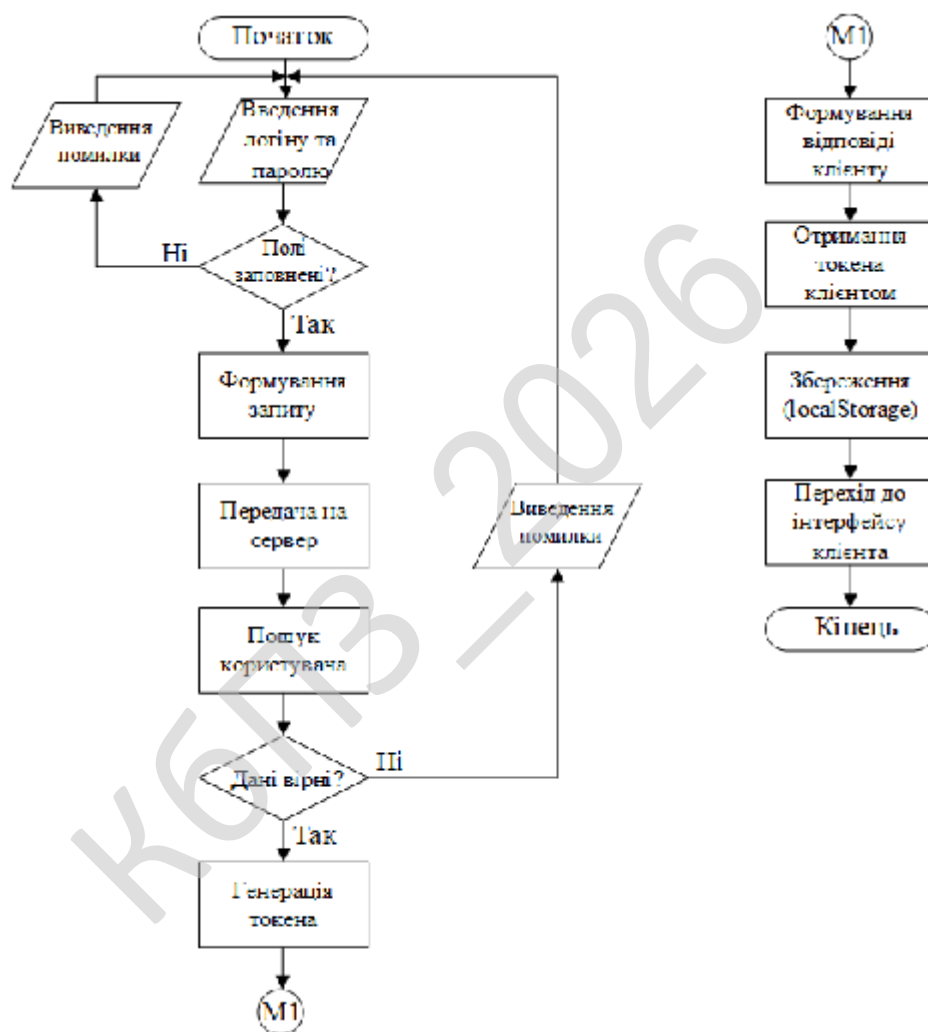


Рисунок 4.1 – Блок-схема процесу авторизації

На рисунку 4.1. наведено процес авторизації користувача у вигляді блок-схеми. Вона має наступні блоки:

1. Початок. Ініціалізація процесу авторизації після відкриття відповідної форми користувачем.

2. Ввід логіну та пароля. Користувач вводить облікові дані через веб інтерфейс. Дані зчитуються з HTML-полів і передаються у програмну логіку.

3. Формування запиту . Після заповнення форми клієнт формує HTTP-запит до сервера. Для реєстрації використовується endpoint /register, а для входу – /login. У тілі запиту передаються логін і пароль у форматі JSON. Цей запит ініціює взаємодію між клієнтською та серверною частиною системи.

4. Передача на сервер. Запит надсилається через мережу на серверну частину системи з використанням API. Сервер отримує дані з req.body, після чого передає їх у функції register() або loginUser() у модулі auth.js. На цьому етапі відбувається первинна обробка введених користувачем даних.

5. Пошук користувача. Сервер виконує перевірку відповідності логіну та пароля даним, що зберігаються. Під час реєстрації перевіряється, чи існує вже користувач з таким логіном у файлі users.json. Під час авторизації перевіряється, чи існує користувач і чи збігається введений логін та пароль із збереженим.

6. Перевірка чи дані вірні. Якщо ні – формується відповідь, що інформує про помилку і процес завершується. Якщо так – виконується наступний блок.

7. Генерація токена. Сервер створює токен (jwt), за допомогою бібліотеки jsonwebtoken, який містить інформацію про користувача, а також встановлюється термін дії (1 година). Цей токен використовується для подальшої ідентифікації користувача при доступі до захищених маршрутів.

8. Збереження токена. Токен одразу повертається клієнту, де зберігається у localStorage (вбудоване сховище браузера). Надалі при кожному запиті клієнт додає токен у заголовок Authorization, а сервер не зберігає його, а лише перевіряє його валідність у authMiddleware через jwt.verify, звіряючи підпис і термін дії. Таким чином, токен існує тільки на стороні клієнта, а сервер виконує лише його перевірку для надання доступу до захищених ресурсів.

9. Перехід до інтерфейсу клієнта. Після успішної авторизації користувач переходить у вікно клієнту, про функції якого поговоримо нижче.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

Алгоритм авторизації забезпечує контроль доступу до функціональних можливостей системи.

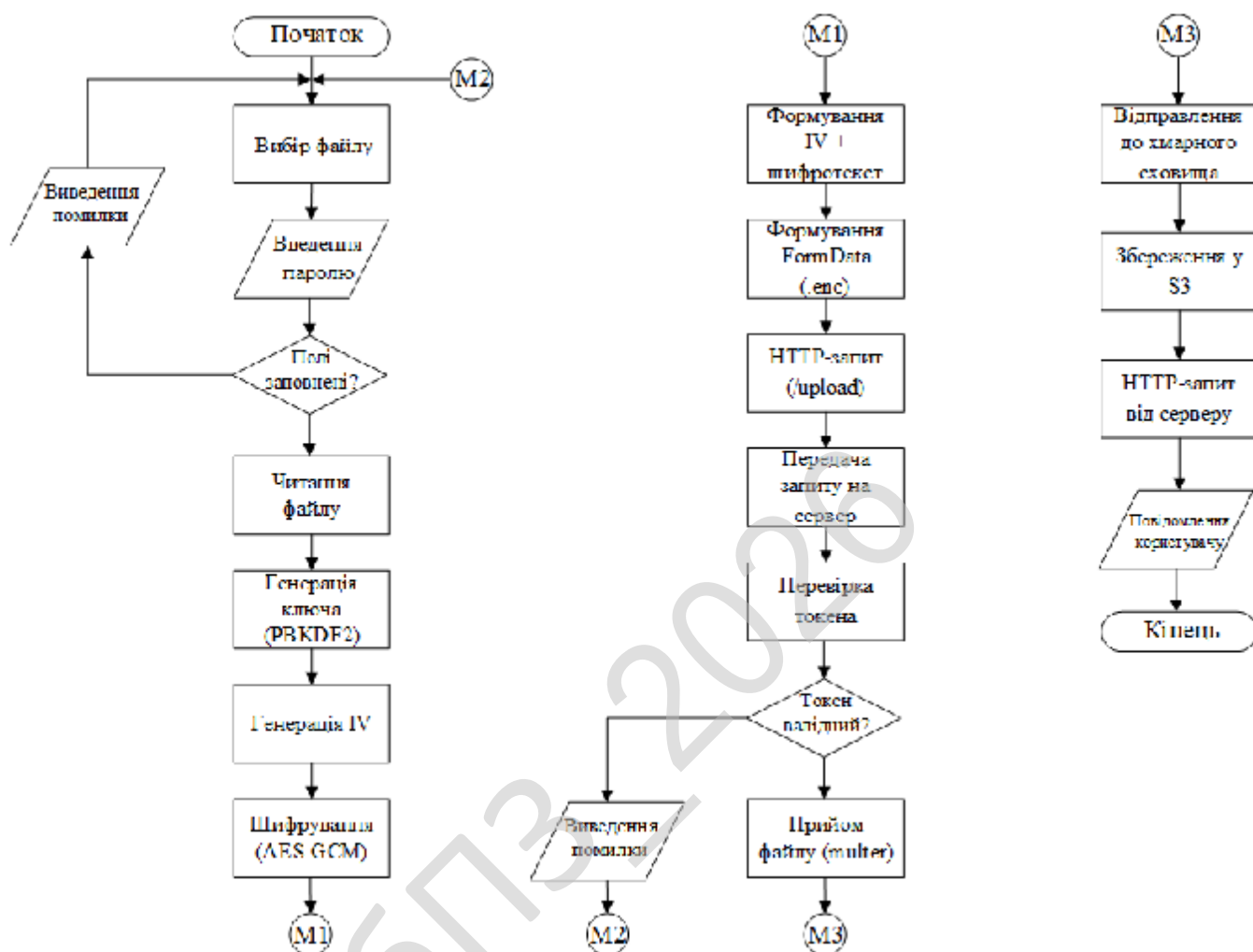


Рисунок 4.2 – Блок-схема процесу завантаження файлу

На рисунку 4.2. наведено процес завантаження файлу користувача до хмарного сховища. Вона має наступні блоки:

1. Початок. Ініціалізація процесу завантаження після проходження авторизації користувачем.
2. Вибір файлу. Користувач вибирає файл для подальшого його шифрування та завантаження на сервер.
3. Введення пароля. Користувач вводить пароль в відповідне поле, цей пароль потім перетвориться на ключ для файлу.

4. Перевірка заповнення полів. Перевіряється чи всі полі заповнені, якщо ні виведеться повідомлення, яке повідомить про це. Якщо всі полі заповнені додаток перейде на наступний крок.

5. Генерація ключа. З пароллю користувача генерується ключ за допомогою PBKDF2 і Web Crypto API.

Алгоритм починається з отримання вхідного параметра password, далі пароль перетворюється в масив байтів (UTF-8), це необхідно для подальших криптографічних операцій.

Створюється базовий криптографічний матеріал. Використовується crypto.subtle.importKey, на основі байтів пароля створюється keyMaterial, він позначається як джерело для алгоритму PBKDF2.

Запускається алгоритму PBKDF2. Застосовується функція deriveKey, яка виконує, формування параметрів PBKDF2: рядок «salt», перетворений у байти, ітерація 100000 повторень хеш функції SHA-256.

Пароль користувача проходить PBKDF2-хешування з salt і 100 000 ітераціями SHA-256, після чого перетворюється у симетричний AES-256 ключ для шифрування та дешифрування даних.

6. Генерація IV. Відповідає за створення випадкового унікального значення, яке використовується під час шифрування. Реалізується за допомогою криптографічно стійкого генератора випадкових чисел crypto.getRandomValues, який створює масив із 12 байтів випадкових значень. Це значення використовується як ініціалізаційний вектор (IV) у режимі шифрування AES-GCM і забезпечує унікальність кожної операції шифрування навіть для однакових вхідних даних. IV не є секретним параметром, однак є критично важливим для безпеки, оскільки запобігає появі однакових шифротекстів при повторному шифруванні одного й того ж файлу. [11]

7. Шифрування даних. На цьому етапі відбувається шифрування файлу користувача за алгоритмом AES-GCM. Алгоритм отримує AES-256 ключ, згенерований через PBKDF2, випадковий 12-байтовий вектор ініціалізації, файл у

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

вигляді ArrayBuffer.

Перед шифруванням виконується створення випадкового вектору ініціалізації IV, що забезпечує унікальність кожного шифрування, щоб навіть однакові дані шифрувались по різному.

8. Формування (IV+шифротекст). Цей алгоритм потрібен для того, щоб сформувати єдиний зашифрований файл, який можна безпечно зберігати або передавати. Оскільки для розшифрування даних в режимі AES-GCM необхідний вектор ініціалізації (IV), його потрібно зберігати разом із зашифрованим контентом.

9. Формування FormData. Після шифрування є два основні елементи: IV – 12-байтовий вектор ініціалізації та результат AES-GCM шифрування (шифротекст + auth tag). Алгоритм виконує формування фінальної структури зашифрованого файлу, виділяючи пам'ять під об'єднаний буфер, записуючи в його початок IV, а після нього – зашифровані дані разом з ідентифікатором достовірності. У результаті створюється самодостатній файл такого формату, який містить усе необхідне для подальшого дешифрування. Зашифрований файл упаковується у формат FormData для відправки на сервер.

10. HTTP-запит на завантаження. створюється HTTP POST-запит (/upload). HTTP-запит на завантаження файлу виконується на клієнтській стороні через fetch з методом POST на endpoint /upload. У цьому запиті передається заголовок Authorization, який містить токен користувача для перевірки доступу, а також тіло запиту у форматі multipart/form-data, де знаходиться файл, вже зашифрований (формату .enc). Після відправки запиту сервер отримує HTTP-повідомлення і запускає відповідний маршрут Express.

11. Перевірка токена. Після натискання кнопки завантаження сервер перевірить токен користувача, для додаткового захисту додатку. Якщо він не вірний то видається повідомлення про помилку, якщо вірний, то перехід до наступного блоку. Виконується authMiddleware, який перевіряє валідність токена користувача і у разі успішної перевірки додає інформацію про користувача в

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		41

req.user

12. Прийом файлу сервером. Прийом файлу на сервері здійснюється за допомогою middleware multer, який обробляє multipart/form-data, витягує файл із запиту та зберігає його в об'єкті req.file.

13. Збереження у хмарі. Процес збереження файлу реалізується у функції uploadFile(user, file) і використовує AWS S3 клієнт для відправки даних у хмарне сховище. Спочатку створюється шлях (key), за яким файл буде збережено в бакеті S3 – «key = `\${user}/\${file.originalname}`», user – ім'я користувача, яким буде названа папка в S3, originalname – оригінальна назва файлу. В результаті отримуємо унікальний шлях у бакеті, наприклад – danya/test.txt.enc.

Створюється команда для завантаження файлу, вона містить назву хмарного сховища, шлях до файлу, бінарний вміст файлу.

Файл передається в AWS S3 через клієнт: встановлюється з'єднання з AWS, файл передається у бакет, дані записуються у хмарне сховище.

14. HTTP-запит від серверу. Після успішного збереження файлу у хмарі, сервер формує HTTP-запит до клієнта і виводить користувачу повідомлення про те, що файл завантажено.

Розглянутий алгоритм завантаження демонструє надійну модель захисту даних за принципом Zero-Knowledge архітектури, де сервер та хмарне сховище оперують виключно зашифрованими масивами інформації, не маючи прямого доступу до початкового вмісту файлів або паролів користувачів. Організація процесу у форматі послідовних криптографічних та мережових абстракцій гарантує цілісність системи на етапі відправки даних.

Для повного розуміння циклу взаємодії з даними необхідно розглянути дзеркальний процес – безпечне вилучення та відновлення файлів із хмари. На рисунку 4.3 зображено блок-схему процесу вивантаження файлу з його дешифруванням. Давайте перейдемо до детального опису цього алгоритму та його функціональних блоків.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

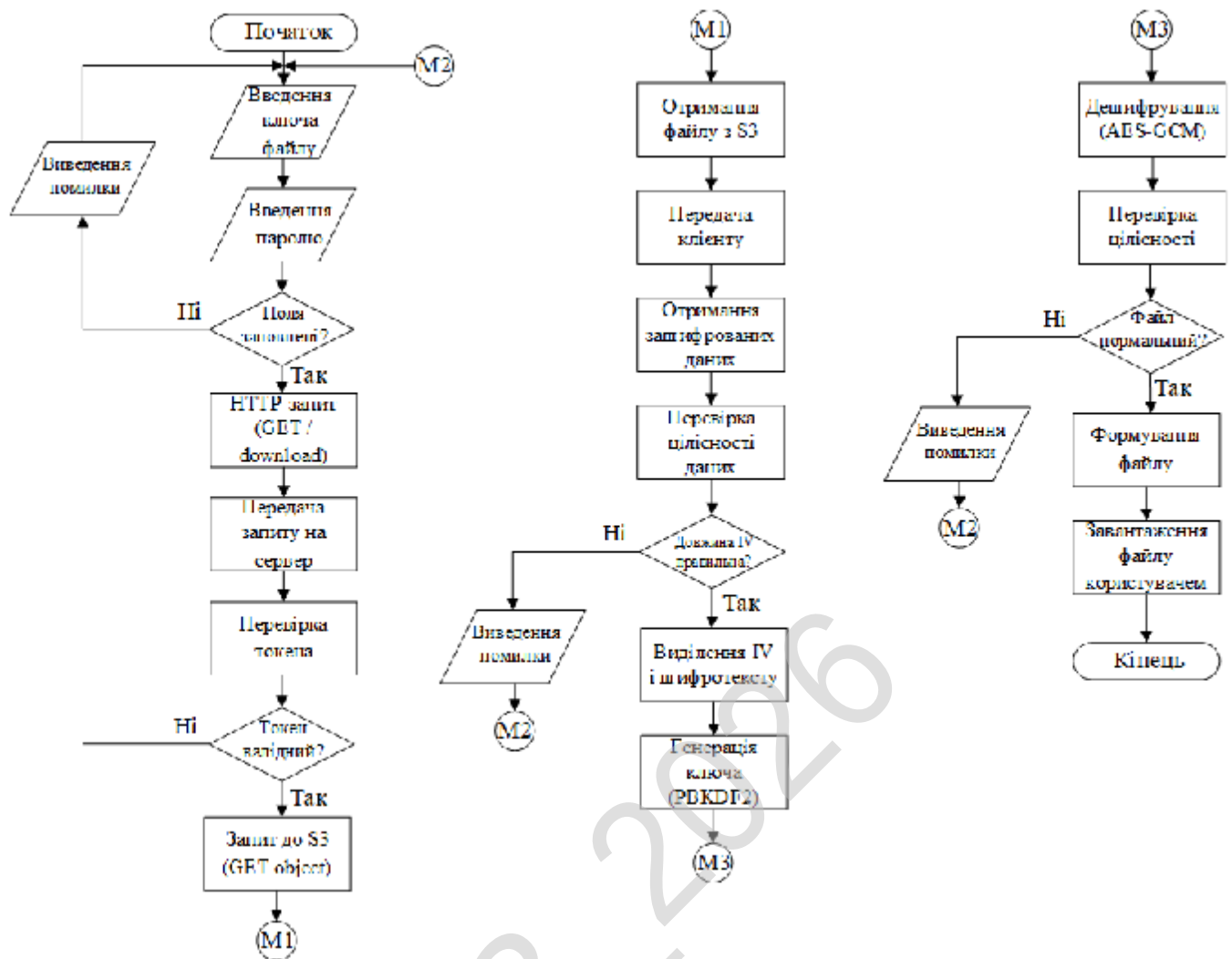


Рисунок 4.3 – Блок-схема процесу вивантаження файлу

На рисунку 4.3. наведено процес вивантаження файлу користувача з хмарного сховища. Він має наступні блоки:

1. Введення ключа файлу. Користувач вводить шлях до файлу та його назву файлу для вивантаження файлу з хмари. Назва каталогу являє собою логін користувача.
2. Введення пароля файлу. Користувач вводить пароль до файлу. Це той самий пароль, який користувач вводив при шифруванні.
3. Перевірка полів. Перевіряється чи всі полі заповнені, якщо ні виведеться повідомлення, яке повідомить про це. Якщо всі полі заповнені додаток перейде на наступний крок.
4. HTTP-запит на завантаження з хмари. HTTP-запит на завантаження

файлу з хмарного сховища виконується в функції download() через fetch з методом GET на endpoint /download. У запиті передається параметр key, який містить шлях до файлу в хмарному сховищі (S3), а також заголовок Authorization, що містить токен користувача для перевірки доступу.

4. Перевірка токена. Перед отриманням файлу з хмарного сховища сервер виконує перевірку токена користувача для забезпечення авторизованого доступу до ресурсу. У процесі перевірки визначається, чи є токен дійсним, чи не прострочений він, а також чи відповідає він зареєстрованому користувачу. Якщо токен не валідний – то процес зупиняється і виводиться повідомлення про помилку. Якщо ні – інформація про користувача додається до об'єкта запиту (req.user), після чого сервер дозволяє подальше виконання запиту і відбувається перехід до наступного блоку.

5. Отримання файлу. Після перевірки авторизації сервер приймає запит на завантаження файлу та отримує з параметра key шлях до об'єкта в хмарному сховищі (S3). Далі викликається функція getFile(key), яка формує запит до AWS S3 через GetObjectCommand. У відповідь хмарне сховище повертає файл у вигляді потокових бінарних даних (stream), які сервер передає назад клієнту без змін. Таким чином сервер виступає посередником між клієнтом і хмарним сховищем, не змінюючи вміст файлу.

6. Передача клієнту. Сервер передає зашифрований файл до клієнтської частини додатку.

7. Перевірка цілісності. Виконує функцію контролю коректності структури зашифрованих даних перед розшифруванням. Оскільки в AES-GCM IV має фіксовану довжину 12 байтів, ця перевірка гарантує, що отриманий файл не пошкоджений, не змінений або сформований неправильно. Під час завантаження зашифрованого файлу спочатку відділяється перші 12 байтів як IV, і система перевіряє, чи відповідає їх розмір очікуваному значенню. Якщо довжина IV відрізняється від 12 байтів, це означає, що структура даних порушена (наприклад, файл був обрізаний, пошкоджений або підмінений), і виводиться помилка для

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		44

користувача. Якщо все добре, то додаток переходить на інший крок.

8. Виділення IV. Після отримання файлу на клієнтській стороні відбувається розділення його структури. Перші 12 байтів бінарного масиву виділяються як ініціалізаційний вектор, а решта даних розглядається як зашифрований контент разом з ідентифікатором достовірності. Це необхідно, оскільки AES-GCM використовує IV для правильного відновлення початкового стану шифрування і подальшого коректного дешифрування даних.

9. Генерація ключа. Паралельно з виділенням IV клієнт генерує криптографічний ключ на основі введеного користувачем пароля. Для цього використовується алгоритм PBKDF2 з хеш-функцією SHA-256, у результаті формується 256-бітний AES-ключ, який буде використаний для розшифрування отриманого файлу. Нова генерація ключа потрібна для забезпечення архітектури додатку zero-knowledge, при якому сервер не зберігає ключі.

10. Дешифрування. Виконується процес дешифрування за допомогою AES-GCM. У функцію `crypto.subtle.decrypt` передається ключ, IV та зашифровані дані разом з authentication tag. Під час дешифрування відбувається не тільки відновлення початкових даних, але й перевірка їх цілісності: якщо дані були змінені або ключ неправильний, процес дешифрування завершується помилкою.

11. Перевірка цілісності файлу. Виконує функцію підтвердження того, що отримані після розшифрування дані є автентичними, не пошкодженими та не були змінені під час зберігання або передачі. Алгоритм використовує автентифікаційний тег (authentication tag), який перевіряється автоматично під час виклику `crypto.subtle.decrypt`. Якщо хоча б один біт зашифрованих даних або IV був змінений, розшифрування не відбудеться і буде виведене відповідне повідомлення користувачу про помилку.

12. Формування файлу. Після успішного дешифрування отримані байти перетворюються у Blob, що дозволяє сформувати файл у браузері. Далі створюється тимчасове посилання (`URL.createObjectURL`), яке автоматично ініціює завантаження файлу користувачу.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

13. Завантаження файлу назад. У результаті користувач отримує відновлений оригінальний файл, який був попередньо зашифрований і збережений у хмарному сховищі.

Реалізація окремих функцій ПЗ

Основна логіка побудована на клієнт-серверній архітектурі, де браузер виконує шифрування та дешифрування, а сервер відповідає за автентифікацію користувачів і передачу файлів до хмарного сховища Amazon S3.

Функція генерації ключа реалізована через PBKDF2 і забезпечує отримання криптографічного AES-256 ключа на основі пароля користувача. Це дозволяє перетворити слабкий текстовий пароль у стійкий ключ шифрування, придатний для симетричної криптографії.

Функція генерації IV створює унікальний 12-байтовий ініціалізаційний вектор за допомогою криптографічно безпечного генератора випадкових чисел. IV додається до кожного шифрованого файлу, оскільки він необхідний для коректного розшифрування та забезпечує унікальність результату навіть при однакових вхідних даних.

Процес шифрування файлу реалізований через алгоритм AES-GCM, де вихідні дані файлу перетворюються у зашифрований бінарний масив. GCM – режим роботи, що широко застосовується в симетричних блокових шифрах, має високу ефективність і продуктивність. Є режимом автентифікованого шифрування (AEAD), надаючи як конфіденційність, так і автентифікацію переданих даних (гарантуючи їх цілісність). [11]

Після шифрування IV з'єднується з шифротекстом, утворюючи єдиний файл, який передається на сервер. Це дозволяє зберігати всю необхідну інформацію для подальшого відновлення даних.

Процес дешифрування відбувається на стороні клієнта: спочатку з отриманого файлу виділяється IV і зашифрований контент, після чого виконується розшифрування за допомогою AES-GCM. У разі невідповідності ключа або пошкодження даних процес автоматично завершується помилкою, що

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		46

забезпечує перевірку цілісності.

Функція авторизації було вирішено додати в додаток для розділення файлів різних користувачів. Являє собою систему реєстрації/входу до основної клієнтської частини додатку. Логін та пароль користувача зберігається на сервері та при авторизації створюється його токен.

Генерація токена та його перевірка у даній системі реалізує механізм автентифікації та контролю доступу користувачів до захищених ресурсів. Після успішного входу (функція /login) сервер перевіряє введені облікові дані та, у разі їх коректності, створює токен доступу(jwt-токен). Цей токен передається клієнту і зберігається у LocalStorage, після чого додається до кожного наступного HTTP-запиту в заголовок Authorization у форматі Bearer. На сервері перевірка токена виконується через проміжне програмне забезпечення (authMiddleware), яке декодує та валідує підпис токена, перевіряє його цілісність. Якщо токен дійсний, у запит додається інформація про користувача (req.user), що дозволяє надалі контролювати доступ до ресурсів, у разі недійсного або відсутнього токена сервер повертає помилку доступу.

4.2 Захист розробленого програмного забезпечення

Програмне забезпечення системи кібербезпеки для шифрування користувацьких даних у хмарних сервісах спрямоване забезпечити користувачу фундаментальні принципи інформаційної безпеки (тріада CIA) – конфіденційність, цілісність та доступність.

Захист конфіденційності користувача досягається шляхом системи авторизації користувачів. Їх розмежування та контроль дій (токен користувача) захищає від отримання ваших файлів іншим користувачем, навіть якщо він знає ваш пароль файлу. Також особливість архітектури zero-knowledge полягає в тому, що ні сервер, ні хмарне сховище не матимуть змоги прочитати ваш файл, так як бачуть його тільки в зашифрованому вигляді.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

Цілісність ваших даних досягається в першу чергу криптографічним алгоритмом AES-GCM із 256-бітним ключем, що забезпечує високий рівень криптографічної стійкості. Ключ шифрування не передається мережею, а генерується локально з пароля користувача за допомогою PBKDF2, що ускладнює атаки перебору. Навіть якщо зломисник отримає доступ до сервера або до хмарного сховища Amazon S3, він не зможе розшифрувати файли без знання пароля. Також AES-GCM має вбудований механізм автентифікації і логіку обробки помилок під час дешифрування – у разі будь-яких змін зашифрованих даних або неправильного ключа процес розшифрування завершується помилкою. Це дозволяє виявляти пошкодження або підміну файлів. Важливим елементом криптографічного захисту є використання ініціалізаційного вектора (IV), який генерується випадковим чином для кожної операції шифрування. Це гарантує унікальність шифротексту навіть у випадку повторного шифрування однакових даних.

Контроль доступності реалізовано через механізм токенів: після автентифікації користувач отримує токен, який використовується для авторизації кожного запиту. Сервер перевіряє дійсність токена та обмежує доступ до ресурсів, дозволяючи працювати лише з файлами, що належать конкретному користувачу. Це запобігає несанкціонованому доступу до чужих даних навіть у межах однієї системи.

Окрему роль відіграє захист мережевої взаємодії. Усі запити між клієнтом, сервером і хмарним середовищем передаються через захищені протоколи (HTTPS), що забезпечує шифрування трафіку та захист від атак типу «людина посередині» (Man-in-the-Middle). Сервер не зберігає конфіденційні дані у відкритому вигляді та виконує лише функції контролю доступу і передачі інформації, що значно зменшує ризики витоку даних.

Таким чином, захист розробленого програмного забезпечення реалізовано як багаторівневу систему, що поєднує криптографічні методи, механізми контролю доступу, перевірку цілісності даних і безпечну мережеву взаємодію.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		48

Такий підхід забезпечує надійний захист інформації навіть у випадку часткової компрометації окремих компонентів системи та відповідає сучасним вимогам до безпеки програмних продуктів.

Перейдемо до ознайомлення з методикою впровадження системи в промислову експлуатацію, де поговоримо яким чином буде відбуватись впровадження системи та наглядно розглянемо роботу програмного забезпечення

К6ПЗ_2026

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Впровадження розробленої системи захищеного зберігання файлів у промислову експлуатацію передбачає поетапний перехід від локального середовища розробки до стабільної хмарної інфраструктури з забезпеченням безперервної роботи, безпеки даних та масштабованості. Ця методика включає в себе наступні етапи:

1. Підготовчий етап. На цьому етапі виконується підготовка середовища для запуску системи, він в себе включає:

- налаштування серверної частини на базі Node.js;
- підготовка конфігураційних файлів (.env) із параметрами доступу до хмарних сервісів Amazon Web Services;
- створення та налаштування хмарного сховища Amazon S3;
- перевірка коректності роботи криптографічних функцій (PBKDF2, AES-GCM).

2. Розгортання серверної інфраструктури. Передбачає підготовку серверу та містить такі аспекти:

- встановлення залежностей Node.js проєкту;
- запуск Express-сервера;
- перевірка API-ендпоінтів (/login, /register, /upload, /download);
- налаштування JWT-авторизації для захисту маршрутів;
- інтеграція з Amazon S3 для зберігання файлів.

3. Розгортання клієнтської частини. Клієнтська частина може бути розміщена на окремому веб-сервері або як статичний сайт, наприклад, через AWS S3 Static Hosting або інший CDN. Вона відповідає за взаємодію з користувачем, шифрування та дешифрування файлів, передачу запитів до API сервера, збереження jwt-токена у локальному сховищі браузера.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

4. Забезпечення безпеки системи. Перед введенням у експлуатацію виконуються заходи безпеки:

- використання HTTPS для шифрування трафіку;
- зберігання паролів у вигляді хешів (bcrypt);
- генерація криптографічних ключів через PBKDF2;
- клієнтське шифрування файлів (AES-GCM);
- ізоляція файлів користувачів у S3 за ім'ям користувача;
- обмеження доступу через JWT-токени.

5. Тестування системи. Тестування розробленої системи проводиться з метою перевірки коректності її роботи, відповідності функціональним вимогам та забезпечення інформаційної безпеки. На функціональному рівні перевіряється робота основних модулів: реєстрація та авторизація користувачів, генерація та перевірка JWT-токенів, завантаження та отримання файлів, а також коректність процесів шифрування і дешифрування даних на стороні клієнта.

Крім функціонального тестування, проводиться перевірка безпеки системи, яка включає спроби доступу без токена, використання неправильних облікових даних, а також перевірку неможливості дешифрування файлів без коректного ключа.

6. Введення в промислову експлуатацію. Введення системи в промислову експлуатацію здійснюється після завершення всіх етапів тестування та усунення виявлених помилок. На цьому етапі виконується розгортання стабільної версії серверної частини у робочому середовищі, а також налаштування взаємодії з хмарним сховищем Amazon Web Services та сервісом Amazon S3 для зберігання зашифрованих файлів користувачів.

7. Подальша підтримка системи. Подальша підтримка системи спрямована на забезпечення її стабільної та безперервної роботи в процесі експлуатації. Вона включає регулярне оновлення програмних компонентів, усунення виявлених помилок, а також оновлення залежностей для підтримки актуального рівня безпеки.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

Перейдемо до опису інструкції програмного додатку для користувача та наглядно подивимось його роботу.

Спочатку нас зустрічає вікно авторизації клієнту (рис. 5.1). Нам пропонується увійти в уже створений акаунт чи створити новий. Для реєстрації треба заповнити поля логіну та паролю, обов'язково обидва поля, якщо хоча б одне пусте, то вам виведеться відповідна помилка (рис 5.2). Також якщо вже є такий користувач то вам видасть помилку (рис 5.3). Якщо все добре користувачу покаже повідомлення про успішну реєстрацію (рис. 5.4).

Рисунок 5.1 – Вікно авторизації клієнту

Рисунок 5.2 – Виведення помилки

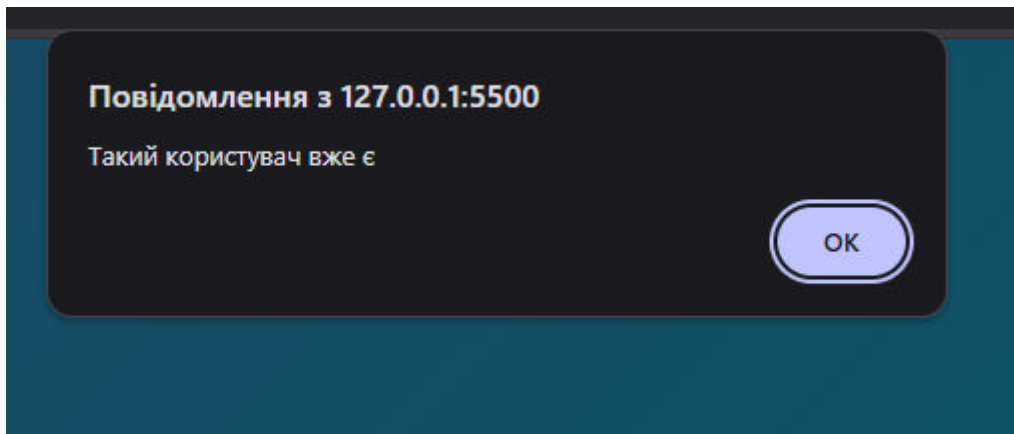


Рисунок 5.3 – Помилка користувача

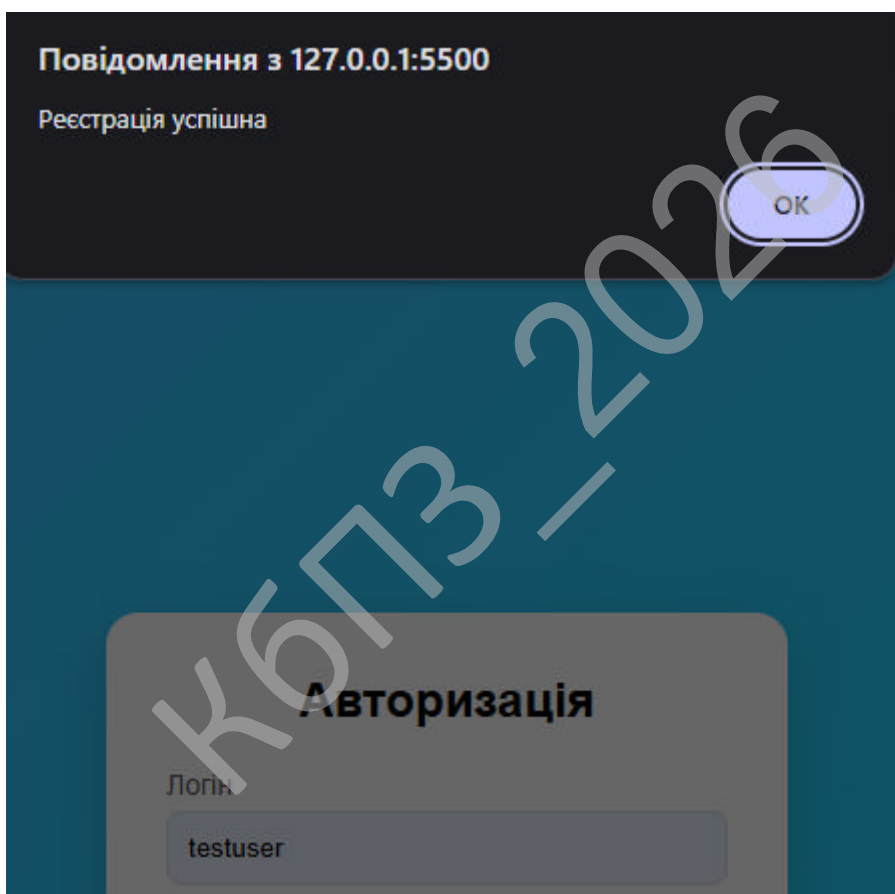


Рисунок 5.4 – Успішна реєстрація

Після реєстрації користувач входить в систему під своїм логіном та паролем, йому виведеться повідомлення про успішний вхід (рис 5.5).

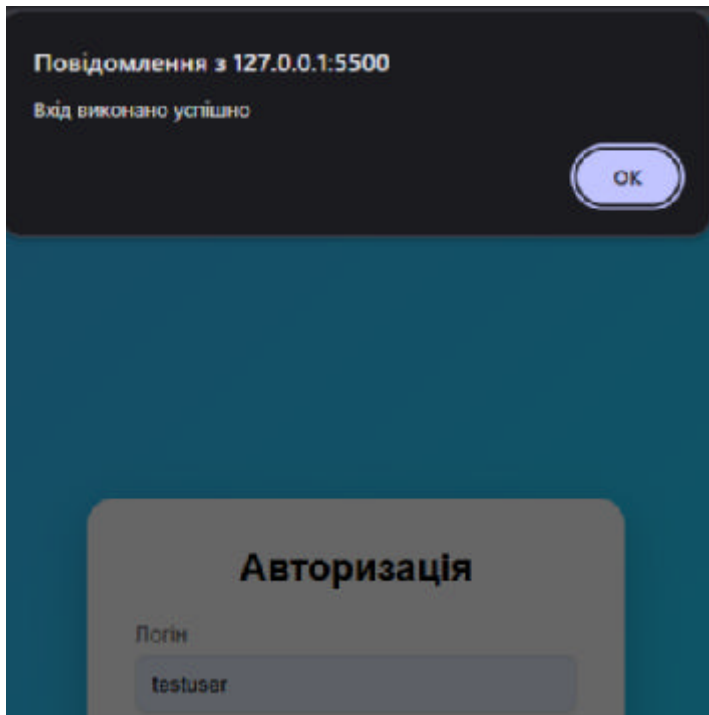


Рисунок 5.5 – Успішний вхід

Після входу користувача зустрічає основне вікно клієнту (рис 5.6).

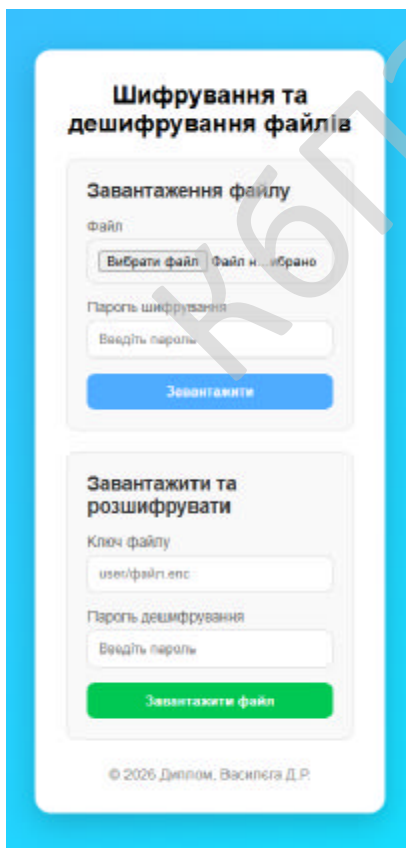


Рисунок 5.6 – Головне вікно клієнту

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		54

Спочатку розглянемо процес шифрування. Користувач обирає файл на своєму пристрої та вводить пароль, який він має запам'ятати. Створимо файл test.txt для наглядної демонстрації роботи додатку, його текст наведено в рисунку 5.7. Вибираємо відповідний файл та вводимо пароль (рис 5.8), цей пароль буде використано при шифруванні.

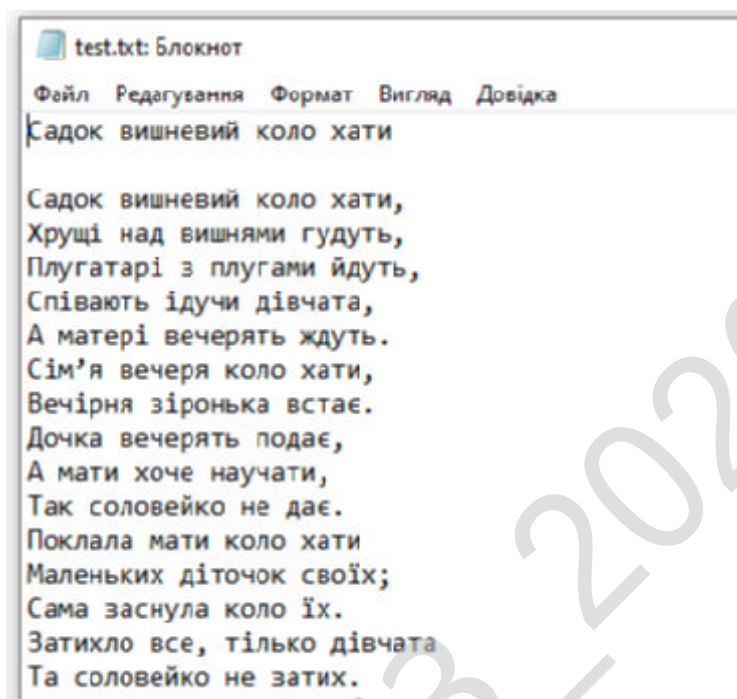


Рисунок 5.7 – Тестовий файл

Рисунок 5.8 – Вибір файлу та ввід пароля

Після того, як користувач натисне кнопку завантажити, файл буде відправлено до сховища, про успіх або невдачу нас повідомить повідомлення користувачу (рис. 5.9).

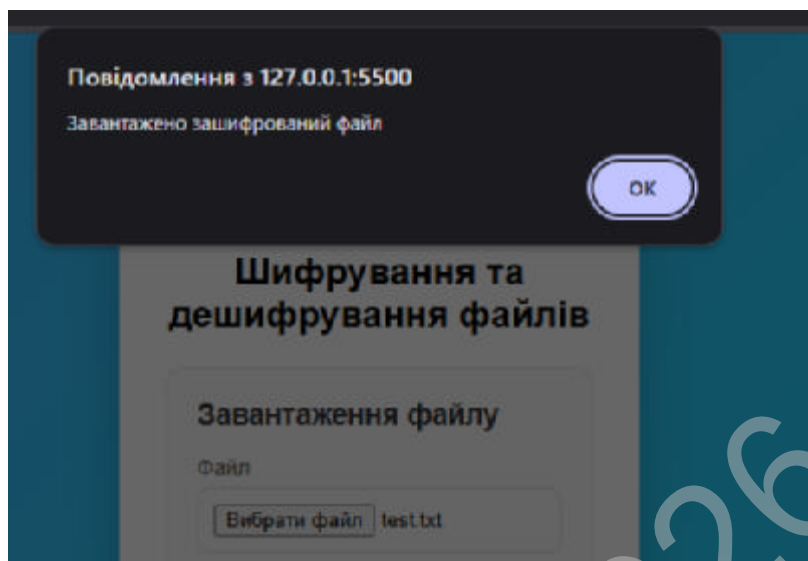


Рисунок 5.9 – Файл завантажено

Переглянемо як виглядають каталоги користувачів в хмарі. На рисунку 5.10 наведено перелік каталогів у хмарі.

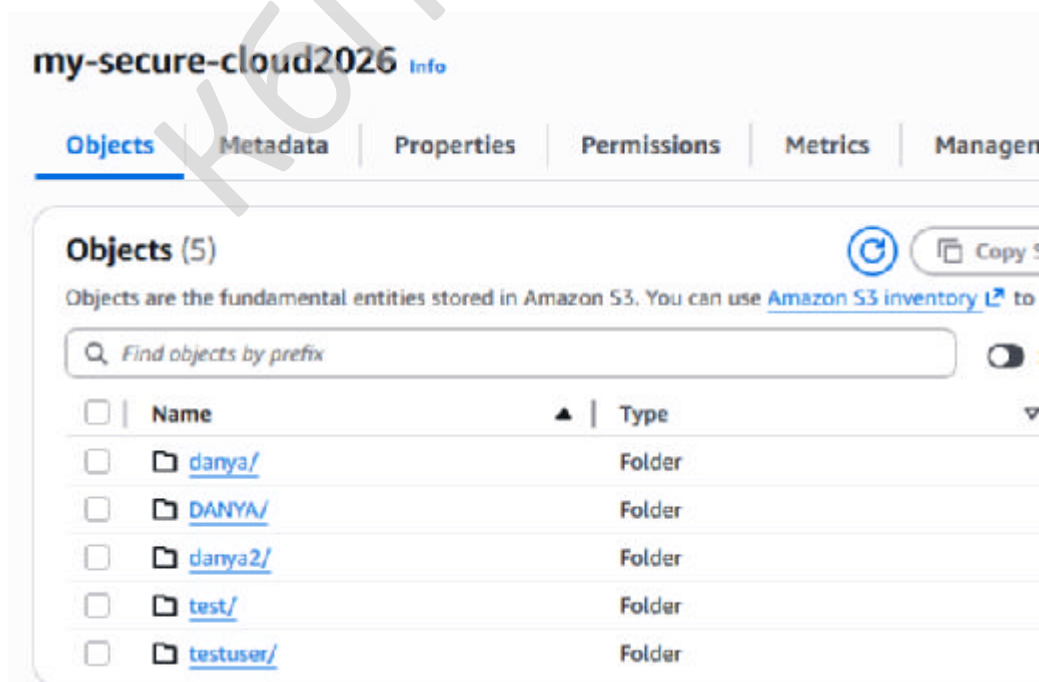


Рисунок 5.10 – Каталоги користувачів

Спробуємо прочитати наш зашифрований файл з хмари. Результат наведено на рисунку 5.11.

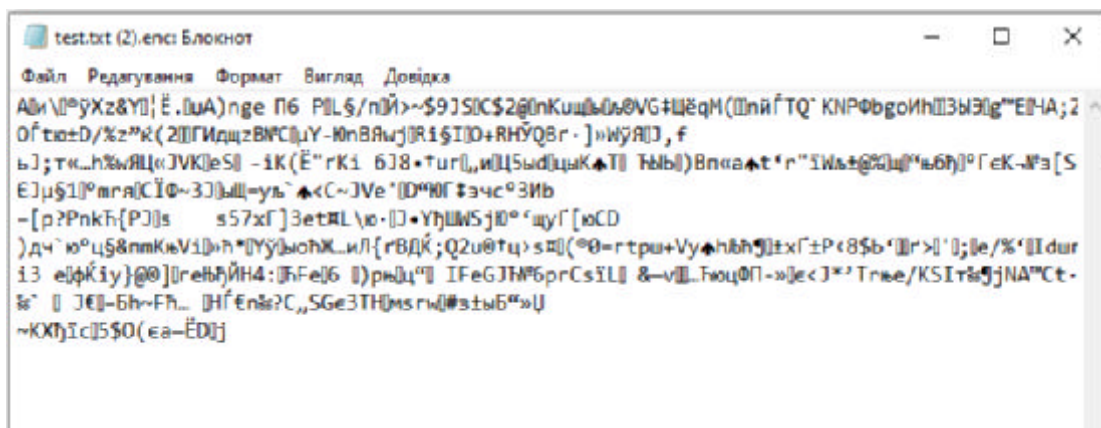


Рисунок 5.11 – Спроба прочитати зашифрований файл

Як ми бачимо все працює добре і ми не можемо переглянути вміст файлу з хмарного сховища. Тепер давайте перейдемо до огляду того, як отримати файл назад. Для цього заповнюємо ключ файлу, це назва та шлях нашого файлу в каталозі хмари. Вона заповнюється по шаблону user/файл.енс, в нашому випадку вводимо testuser/test.txt.enc, після чого вводимо пароль дешифрування(це той пароль, який ми водили при шифруванні). Приклад заповнених даних наведено на рисунку 5.12.

Рисунок 5.12 – Заповнені дані для дешифрування

Після натискання кнопки почнеться процес дешифрування і користувачу надасться змога завантажити розшифрований файл на свій ПК (рис. 5.13). У разі, якщо ви ввели не вірний пароль або файл пошкоджений вам видасть про це повідомлення (рис 5.14).

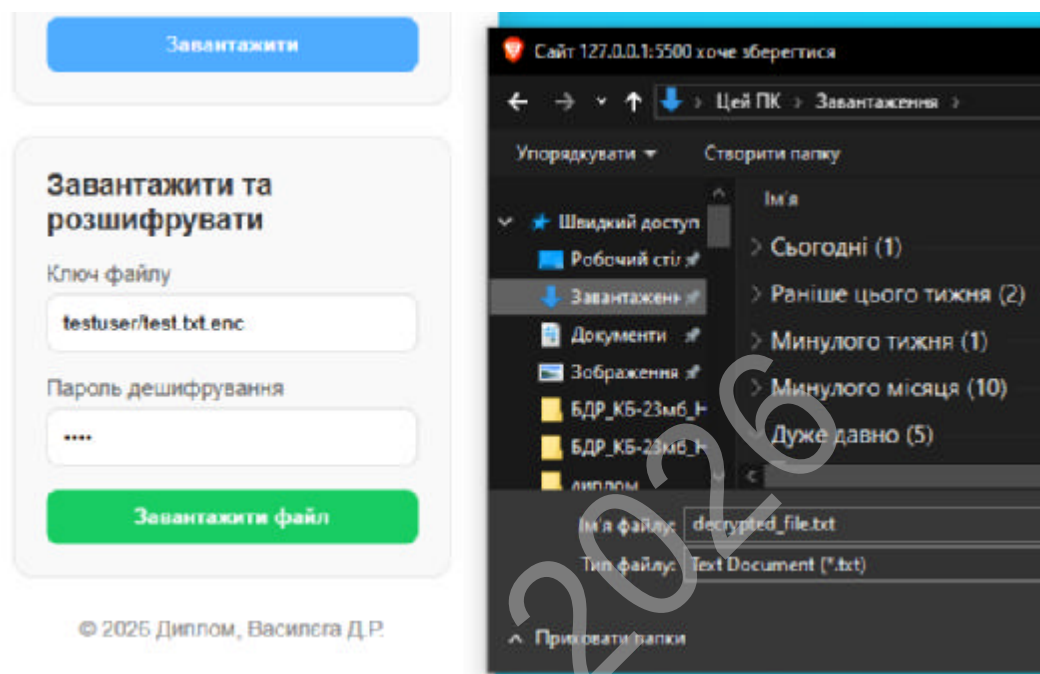


Рисунок 5.13 – Завантаження файлу

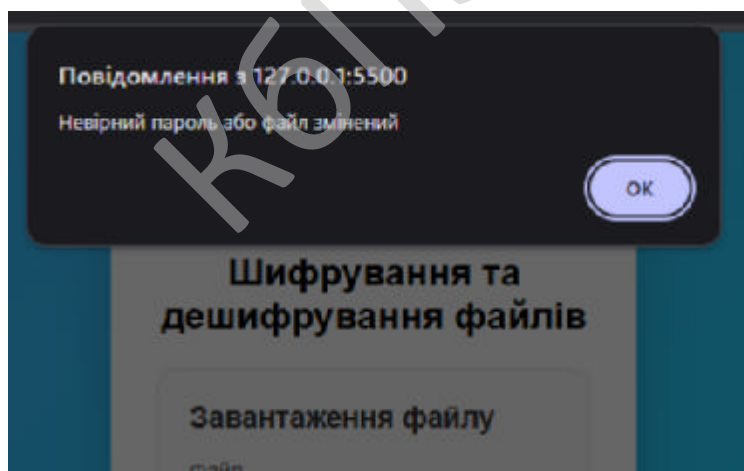


Рисунок 5.14 – Виведення помилки

Спробуємо прочитати розшифрований файл, щоб перевірити результат. На рисунку 5.15 наведений текст розшифрованого файлу.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		58

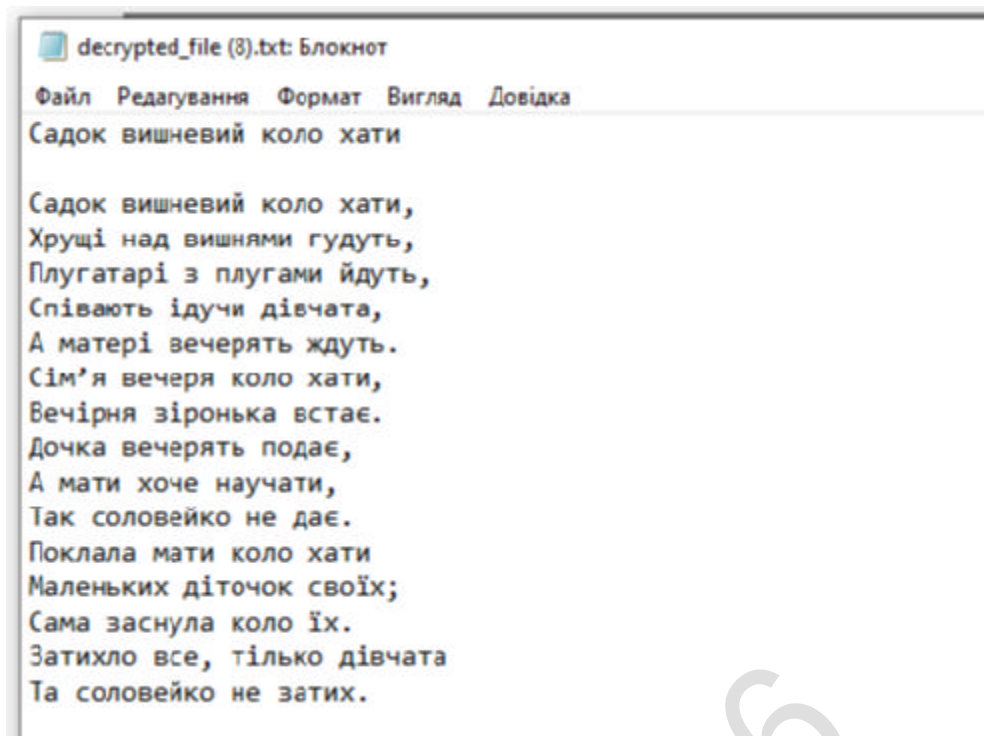


Рисунок 5.15 – Розшифрований текст

Як бачимо результат збігається, отже все працює правильно. Додаток пройшов тестування та готовий до впровадження в експлуатацію. Перейдемо до основних висновків зроблених під час роботи.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		59

6 ОСНОВНІ ВИСНОВКИ

У ході виконання дипломної роботи було розроблено програмну систему для безпечного зберігання файлів у хмарному середовищі, яка забезпечує безпеку даних шляхом їх попереднього шифрування на стороні клієнта.

У процесі роботи було проаналізовано сучасні підходи до захисту інформації та обґрунтовано вибір криптографічних алгоритмів. Реалізовано механізм генерації ключа на основі пароля користувача із застосуванням алгоритму PBKDF2, а також шифрування та дешифрування даних за допомогою алгоритму AES-GCM, що забезпечує не лише конфіденційність, але й контроль цілісності даних.

Було розроблено клієнтську частину системи, яка виконує криптографічні операції, а також серверну частину, що забезпечує обробку запитів, автентифікацію користувачів та взаємодію з хмарним сховищем. При цьому сервер не має доступу до відкритих даних, що підвищує загальний рівень безпеки системи.

Реалізовано механізми завантаження та вивантаження файлів, які включають перевірку доступу, обробку даних та їх передачу у зашифрованому вигляді. Проведені експериментальні дослідження підтвердили працездатність системи та коректність реалізованих алгоритмів.

Отримані результати свідчать про доцільність використання клієнтського шифрування для захисту даних у хмарних середовищах. Розроблена система може бути використана як основа для подальшого розвитку програмних рішень у сфері інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2025 Thales Cloud Security Study. Thales. URL: <https://cpl.thalesgroup.com/cloud-security-research#form-download>
2. Колесніков Д. Що таке кібербезпека?. Brainlab. URL: <https://brainlab.com.ua/uk/blog-uk/shho-take-kiberbezpeka>
3. Різновиди загроз, яким протидіє кібербезпека на підприємствах. Дія. URL: https://business.diia.gov.ua/entrepreneur-handbook/item/riznovidi_zagroz_yakim_protidie_kiberbezpeka_na_pidpriemstvah
4. Головні кіберзагрози для хмар: як захистити свої дані у 2025 році. colobridge. URL: <https://blog.colobridge.net/uk/2022/05/top-cloud-cyber-threats-ua/>
5. Cloud Security 2025: Як експерти зміцнюють кіберзахист бізнесу, коли дані переміщуються в хмару. robotdreams. URL: <https://robotdreams.cc/uk/blog/820-nauposhyrenishi-vyklyky-dlya-khmarnoyi-bezpeky>
6. Хмарні сховища. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Хмарні_сховища
7. Топуа. Що таке хмарне сховище?. Ucloud. URL: <https://ucloud.ua/shho-take-hmarne-shovyshhe/>
8. Types of Cloud Storage: A Complete Guide to Deployment Models and Data Organization. Hivenet. URL: <https://www.hivenet.com/post/types-of-cloud-storage-a-complete-guide-to-deployment-models-and-data-organization>
9. Data encryption as a method of protecting personal data in a cloud environment. Вісник Черкаського державного технологічного університету. 2025. Vol. 29, no. 3. URL: <https://surl.li/xyqzbc>
10. Galois/Counter Mode. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Galois/Counter_Mode
11. Режими блокового шифрування. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Режими_блокового_шифрування

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

12. Хмарна безпека: чи згущуються хмари над захистом даних?. ТОВ «ВИДАВНИЧИЙ ДІМ «МЕДІА-ДК». URL: https://biz.nv.ua/ukr/kibervoiny_i_biznes/hmarna-bezpeku-zgushchujutsja-chi-hmari-nad-zahistom-daniv-2331588.html

13. CSA Security Guidance for Critical Areas of Focus in Cloud Computing. Cloud Security Alliance. URL: <https://cloudsecurityalliance.org/research/guidance>.

14. Node.js v26.1.0 documentation. Node.js. URL: <https://nodejs.org/uk>

15. Web Crypto API. MDN. URL: https://mdn2.netlify.app/en-us/docs/web/api/web_crypto_api/

16. OpenJs. Fast, unopinionated, minimalist web framework for Node.js. URL: <https://expressjs.com/>

17. AWS Documentation/Security in Amazon S3. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/security.html>

18. Google Cloud /Cloud Storage. URL: <https://cloud.google.com/storage/docs/encryption/>

19. Microsoft Azure/Azure Storage. URL: <https://docs.microsoft.com/en-us/azure/storage/common/storage-service-encryption>

20. Бурячок В. Л., Киричок Р. В., Складанний П. М. Основи інформаційної та кібернетичної безпеки. – Київ, 2019. 320с.

21. Самойлов І. В., Матійко А. А., Сторчак А. С. Криптографія. – Київ: КПІ ім. Ігоря Сікорського, 2023. 372с.

22. Методичні рекомендації до виконання й захисту випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти для здобувачів вищої освіти за спеціальністю 125 “Кібербезпека” / [уклад. О.А. Смірнов, Є.В. Мелешко, К.О. Буравченко, С.А. Смірнов, Н.М. Якименко, О.П. Доренський, Т.В. Смірнова, А.С. Коваленко]; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2025. 65 с.

23. Документація по Visual Studio. Microsoft. URL: <https://learn.microsoft.com/uk-ua/visualstudio/windows/?view=visualstudio>

24. bcrypt. Вікіпедія. URL: <https://en.wikipedia.org/wiki/Bcrypt>

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

25. Introduction to JSON Web Tokens. JWT Debugger. URL: <https://www.jwt.io/introduction#what-is-json-web-token>
26. Multer. GitHub. URL: <https://github.com/expressjs/multer>
27. Zero-knowledge proofs. Wikipedia. URL: https://en.wikipedia.org/wiki/Zero-knowledge_proof
28. Допомога та навчання з Visio. Microsoft. URL: <https://support.microsoft.com/uk-UA/visio>
29. Amazon Simple Storage Service Documentation. AWS. URL: <https://docs.aws.amazon.com/s3/>
30. Mirza N. Хмари та кібербезпека. Linkedin. URL: <https://ua.linkedin.com/pulse/clouds-cyber-security-essential-4-pillars-naeem-mirza?tl=uk>
31. Самойлов І. В., Матійко А. А., Сторчак А. С. Криптографія. Київ: КПП ім. Ігоря Сікорського, 2023. с.9-43
32. Л.Я Глинчук. КРИПТОЛОГІЯ : Навч.-метод. посіб. Луцьк : Вежа-Друк, 2012. 186 с..
33. Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселичник, А. П. Бондарєв, С. С. Войтусік, А. Я. Горпенюк, О. А. Немкова, І. М. Журавель, Б. М. Березюк, Є. І. Яковенко, В. І. Отенко, 1. Я. Тишик. Інформаційна безпека. Львів: Видавництво Львівської політехніки 2019. с.581
34. Василь Олексюк. ОСНОВИ ХМАРНИХ ТЕХНОЛОГІЙ : Навч.-метод. посіб. Тернопіль : Гарнітура Times, 2018. 156 с.
35. Ю. Є. Яремчук О. В. Салієва І. О. Бондаренко. ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ : Електрон. навч. посіб. Вінниця : ВНТУ, 2024. 140 с.
36. О. С. Бунке. СЕРВЕРНІ WEB-ТЕХНОЛОГІЇ : Навч. посіб. Київ : КПП ім. Ігоря Сікорського, 2023. 109 с.
37. ДСТУ ISO/IEC 27018:2019 Інформаційні технології. Методи захисту. Кодекс ustalеної практики для захисту персональної ідентифікаційної

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

інформації (PII) у загальнодоступних хмарах, що діють як процесори PII (ISO/IEC 27018:2019, IDT).

38. ДСТУ 3396 0-96 Захист інформації. Технічний захист інформації. Основні положення;

38. Усік П.С., Смірнова Т.В., Бурмак Ю.А., Улічев О.С., Доренський О.П., «Стійка функція шифрування удосконаленого модуля криптографічного захисту інформації в інформаційно-комунікаційних системах» Кібербезпека: освіта, наука, техніка. № 1(13). С. 183-201.

39. Вітер М.Б. Використання Хмарних технологій у системі інформаційної взаємодії державних органів. Науковий вісник НЛТУ України. 2014. с.340-347

40. Юдін О.К., Зюбіна Р.В. Нормативно-правові аспекти використання Хмарних технологій. Наукоємні технології. 2014. с. 303-306.

41. Чігіна Н.В. Поняття та основні правові проблеми упорядкування відносин у сфері Хмарних технологій. Правова інформатика. 2014. С 17-24

42. Криновський Р. Принципи правового регулювання використання Хмарних технологій для обробки персональних даних. URL: <https://ideas.repec.org/a/pos/journal/60-2.html>

43. Брижко В.М. Приватність даних у Хмарних технологіях. Інформація і право. с. 47-59.

44. Vacca J. R. Cloud Computing Security: Foundations and Challenges. 2nd ed. CRC Press, 2020. 574 p.

45. Євсєєв С. П., Король О. Г., Гололобов О. В. Захист інформації в інформаційно-комунікаційних системах: навчальний посібник. Харків: ХНЕУ ім. С. Кузнеця, 2018. с.350.

46. AWS Security Best Practices / Amazon Web Services. URL: <https://aws.amazon.com/whitepapers/aws-security-best-practices/>

47. Amazon S3 Developer Guide / Amazon Web Services. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

48. Mondal, A., Paul, S., Goswami, R.T., Nath, S. Cloud computing security

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		64

issues & challenges. 2020. с 22-24

49. Abha, S.; Bhanali, M. Cloud computing security using AES algorithm. 2013. с. 19-23.

50. Rani, S.; Gangal, A. Cloud security with encryption using hybrid algorithm and secured endpoints. 2012. с. 4302– 4304

51. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 1(67). С. 84-89.

52. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». VII міжнародна науково-практична конференція “Інформаційна безпека та комп’ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. - Кропивницький: ЦНТУ. - 2023. - С. 26.

					ВКРБ-125.26.0033.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		65