НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «_Концептуальна_модель_та_фрагменти_програмного забезпечення_захищеного_десктопного_месенджеру»_____

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи Асистент	_____ (підпис)	**Ярослав_Гордій**
Консультант к.т.н., доцент	_____ (підпис)	**Белла_ГОЛУБ**
Виконав	Масло _____ (підпис)	**Сергій_Масло**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«_6_» __січня_____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

_____Масла_Сергія_Вячеславовича_____
(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«концептуальна модель та фрагменти програмного забезпечення захищеного десктопного месенджеру»

затверджена наказом від «_6_» _____січня_____2026 р. № _46_ «С»
Термін подання завершеної роботи на кафедру «25» __травня_____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): забезпечення захищеного обміну повідомленнями та файлами в десктопному месенджері з наскрізним шифруванням.
Перелік питань, що підлягають дослідженню:

- Аналіз вимог та моделі загроз для захищеного десктопного месенджера.
- Розробка концептуальної моделі системи
- Проєктування фрагментів програмного забезпечення клієнтської та серверної частин
- Реалізація та тестування прототипних фрагментів ПЗ, оцінка відповідності вимогам безпеки

Дата видачі завдання «_6_» ____Січня_____2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Ярослав Гордій**

Завдання взяв до виконання _____ (підпис) **Сергій МАСЛО**

Зміс

т

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Позначення	**Розшифрування**
API	Програмний інтерфейс застосунку
AES-GCM	Криптографічний інструмент
C#	Мова програмування
CI/CD	Підхід до автоматизованої інтеграції та розгортання програмного забезпечення
DPAPI / ProtectedData	Windows-орієнтований механізм захисту локального ключового матеріалу
E2EE / end-to-end encryption	Наскрізне шифрування
RAM	Оперативна пам’ять
RSA-PSS/SHA-256	Криптографічний інструмент
SHA-256	Криптографічна хеш-функція
SQL	Мова структурованих запитів
SQLite	Локальна база даних
UML	Уніфікована мова моделювання
WinForms	Тип десктопного застосунку
ZIP-пакет	Орієнтовний архівний склад повної версії програмної системи
БД	База даних
ОС	Операційна система
ПК	Персональний комп’ютер
ПЗ	Програмне забезпечення
СУБД	Система управління базами даних

ВСТУП

У сучасних інформаційних системах обмін повідомленнями та файлами є одним із базових способів цифрової взаємодії між користувачами. Водночас для таких систем важливими залишаються питання захисту змісту повідомлень, контролю доступу до даних, зберігання локального контенту, обробки службових станів і визначення ролі серверного контуру. Особливої уваги потребують десктопні застосунки, у яких значна частина даних пов’язана з пристроєм користувача, локальним сховищем і ключовим матеріалом.

Актуальність роботи зумовлена потребою в моделюванні захищеного десктопного месенджера, який поєднує персональний обмін повідомленнями й файлами, локальне зберігання основного контенту, службову роль сервера та контроль активного пристрою. У межах такої моделі важливо не лише описати функції обміну, а й визначити обмеження системи, ризики, правила доступу, життєві цикли повідомлень і межі глобального видалення.

Метою дипломної роботи є розроблення концептуальної моделі та розгляд фрагментів програмного забезпечення захищеного десктопного месенджера.

Об’єктом дослідження є процес захищеного обміну повідомленнями та файлами в десктопному месенджері.

Предметом дослідження є концептуальна модель, інформаційні процеси, контури зберігання, правила доступу та окремі програмні фрагменти захищеного десктопного месенджера.

Для досягнення мети в роботі поставлено такі завдання:

1. дослідити предметну область захищених десктопних месенджерів;
2. сформувати вимоги до захищеного месенджера;
3. проаналізувати ризики й обмеження запропонованої моделі;
4. оглянути існуючі рішення у сфері захищених месенджерів;
5. побудувати концептуальну модель системи;

6. визначити ролі, сутності, зв’язки, інформаційні процеси, життєві цикли та контури захищеності;

7. розглянути фрагменти програмного забезпечення для прив’язки пристрою, обміну повідомленнями та локального збереження файлових даних.

У роботі використано системний аналіз предметної області, порівняльний аналіз існуючих рішень, концептуальне моделювання, логічне моделювання даних, UML-моделювання та фрагментарну програмну реалізацію мовою C#.

Практична частина роботи обмежується окремими програмними фрагментами. У роботі не підтверджується створення повної промислової системи, завершеної серверної інфраструктури, фізичної бази даних, повного криптографічного протоколу або готового інсталяційного пакета. Програмні фрагменти мають демонстраційний характер і використовуються для пояснення ключових механізмів моделі.

Дипломна робота складається з трьох розділів. У першому розділі виконано системний аналіз предметної області, визначено вимоги, ризики, обмеження, розглянуто існуючі рішення та сформульовано постановку завдання. У другому розділі побудовано концептуальну модель захищеного десктопного месенджера, визначено ролі, сутності, зв’язки, процеси, життєві цикли та модель захищеності. У третьому розділі розглянуто фрагменти програмного забезпечення, демонстраційну перевірку, вимоги до середовища виконання та орієнтовний склад ZIP-пакета повної версії програмної системи

РОЗДІЛ 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

У цьому розділі виконується системний аналіз предметної області захищених десктопних месенджерів і визначаються межі подальшого моделювання. Розгляд починається з особливостей обміну повідомленнями й файлами в десктопному форматі, після чого уточнюються вимоги, ризики, обмеження та підходи, реалізовані в існуючих рішеннях. Завершується розділ постановкою завдання, яка фіксує клас системи, основні дані, автоматизовані операції та очікувані результати для побудови концептуальної моделі захищеного месенджера.

1.1 Особливості предметної області захищених месенджерів

У межах цієї роботи месенджер розглядається як прикладна інформаційна система, що поєднує передавання повідомлень, обмін файлами, роботу з обліковими записами, підтримку сеансу користувача та обробку службових станів. Для захищеного месенджера важливий не лише факт комунікації, а й спосіб захисту змісту повідомлень, місце зберігання даних, межі доступу до них і роль серверної частини.

Обрана модель стосується захищеного десктопного месенджера. Десктопний формат важливий тому, що основний зміст листування пов’язується з пристроєм користувача: на клієнтській стороні формується локальна історія повідомлень, виконуються операції з файлами та підтримується локальне прикладне сховище. Отже, клієнтський застосунок є не лише інтерфейсом, а й частиною середовища зберігання та захисту основного контенту.

Предметна область охоплює створення діалогу, надсилання й отримання повідомлень, передавання файлів, криптографічну обробку повідомлень і файлів, формування квитанцій, локальне збереження контенту, редагування,

локальне або глобальне видалення, зміну активного пристрою та службову обробку зашифрованих недоставлених повідомлень. Повідомлення в такій системі не є лише текстовим записом: з ним пов’язані стани доставки, службові ознаки обробки, файлові вкладення, квитанції та сценарії видалення.

Особливістю запропонованої моделі є обмежена службова роль сервера. Серверний контур виконує маршрутизацію, перевірку службового формату пакета, забезпечення доставки та тимчасове зберігання зашифрованих недоставлених повідомлень, але не розглядається як основне сховище відкритого змісту листування. У межах цієї моделі сервер не повинен мати доступу до відкритого змісту повідомлень.

Базова модель орієнтована на персональний обмін один-на-один. Групові чати, складна багатостороння взаємодія та синхронізація між багатьма пристроями не входять до основного розгляду. Окреме значення має активний пристрій: користувач працює через один чинний клієнтський контур доступу, а під час зміни пристрою попередній пристрій має втратити активний статус. Ключовий матеріал має бути пов’язаний з апаратно або системно захищеним сховищем операційної системи; конкретна технологія такого сховища в цьому розділі не визначається.

Корпоративний контур має допоміжний характер. Він може використовуватися для резервного копіювання, аудиту, адміністрування або відновлення доступу, але не змінює основний фокус базової моделі: локальне зберігання основного контенту, службову роль сервера, контроль активного пристрою та персональний формат обміну повідомленнями.

1.2 Аналіз вимог до захищеного месенджера

Вимоги до захищеного десктопного месенджера визначають межі майбутньої моделі. Вони показують, які дії має підтримувати система, які дані потребують захисту, які службові процеси потрібні для контрольованої роботи і які технічні параметри обмежують застосування системи. У межах цієї роботи

вимоги стосуються саме захищеного десктопного застосунку з локальним зберіганням основного контенту, службовою роллю сервера та одним активним пристроєм.

Вимоги до системи доцільно поділити на функціональні, безпекові, нефункціональні та службові. Такий поділ узагальнено в табл. 1.1.

Таблиця 1.1

Основні вимоги до захищеного десктопного месенджера

№	Група вимог	Зміст вимог	Значення для моделі
1	Функціональні	Реєстрація користувача, вхід, створення діалогу один-на-один, надсилання й отримання повідомлень, передавання файлів, редагування, локальне та глобальне видалення, квитанції доставки й прочитання	Формують прикладну основу роботи месенджера
2	Безпекові	Передавання повідомлень у зашифрованому вигляді, локальне зберігання основного контенту у зашифрованому вигляді, захищене зберігання ключового матеріалу, один активний пристрій	Визначають захищений режим роботи системи
3	Нефункціональні	Первинний цикл доставки, офлайн-зберігання зашифрованих недоставлених повідомлень, обмеження розміру файла, кількість локально доступних чатів, резервні копії	Задають технічні межі запропонованої моделі
4	Службові	Зміна активного пристрою, деактивація попереднього пристрою, аудит, резервування, відновлення доступу, адміністрування в корпоративному контурі	Підтримують службові процеси системи

Функціональні вимоги охоплюють базову прикладну логіку: реєстрацію, вхід, створення діалогу, обмін повідомленнями й файлами, редагування, локальне та глобальне видалення, а також квитанції доставки й прочитання. Безпекові вимоги визначають захищений характер системи: повідомлення

передаються у зашифрованому вигляді, основний контент зберігається локально у зашифрованому вигляді, сервер не повинен мати доступу до відкритого змісту повідомлень, основний контент зберігається локально у зашифрованому вигляді [16], сервер не повинен мати доступу до відкритого змісту повідомлень, а доступ користувача обмежується одним активним пристроєм.

Нефункціональні вимоги задають технічні межі моделі. Первинний цикл доставки передбачає до трьох повторних спроб у межах 10 секунд. Якщо адресат перебуває офлайн, серверний службовий контур може тимчасово зберігати зашифроване недоставлене повідомлення до моменту доставки, але не довше шести місяців. Також задано максимальний розмір файла 50 МБ і до 500 локально доступних чатів. Точна числова межа кількості користувачів корпоративної інсталяції в наданих матеріалах не зафіксована. Числові параметри подано в табл. 1.2.

Таблиця 1.2

Числові параметри запропонованої моделі

Параметр	Значення в моделі
Кількість повторних спроб доставки	до 3 спроб
Час первинного циклу доставки	до 10 секунд
Строк офлайн-зберігання зашифрованого недоставленого повідомлення	до 6 місяців
Максимальний розмір файла	50 МБ
Кількість локально доступних чатів	до 500 чатів
Кількість користувачів корпоративної інсталяції	точна числова межа не зафіксована

Наведені числові параметри є межами запропонованої моделі. Зовнішнє обґрунтування цих значень у наявних матеріалах не визначено, тому вони не подаються як загальні стандарти для захищених месенджерів.

Службові вимоги охоплюють зміну активного пристрою, деактивацію попереднього пристрою, аудит критичних подій, резервування та відновлення доступу. У корпоративному контурі вони можуть також охоплювати

адміністрування користувачів і пристроїв, політики резервування, адміністративні сповіщення та базову звітність, але цей контур залишається допоміжним.

Окремо потрібний контроль помилок: система має відхиляти некоректні запити, операції з неактивного пристрою, неправильні службові поля пакета, повторну обробку вже отриманого повідомлення, некоректний файл або файл, що перевищує встановлене обмеження. Вимоги також задають основу для формалізації даних користувача, пристрою, діалогу, повідомлення, файла, квитанції та службового запиту, яка подається в постановці завдання.

Отже, вимоги фіксують не лише перелік функцій, а й межі майбутньої моделі: персональний обмін повідомленнями, локальне зберігання основного контенту, службову доставку через серверний контур, контроль активного пристрою, обмеження роботи з файлами, резервування й аудит.

1.3 Аналіз ризиків і обмежень системи

Ризики й обмеження захищеного десктопного месенджера пов’язані не лише із зовнішніми загрозами, а й з архітектурними рішеннями моделі: локальним зберіганням основного контенту, службовою роллю сервера, одним активним пристроєм і тимчасовим зберіганням зашифрованих недоставлених повідомлень.

Основні ризики стосуються передавання повідомлень, локального сховища, активного пристрою, серверного службового контуру, резервних копій і процедур відновлення доступу. Такий поділ ризиків узгоджується із загальним підходом до визначення безпекових контролів і ризиків інформаційних систем [18]. Вони узагальнені в табл. 1.3.

Таблиця 1.3

Основні ризики захищеного десктопного месенджера

№	Ризик	Причина	Можливий наслідок	Механізм зменшення
1	Перехоплення або підміна пакета	Передавання повідомлення через службовий серверний контур	Порушення цілісності або коректності доставки повідомлення	Зашифроване передавання, службова перевірка пакета, перевірка цілісності службового пакета
2	Повторна обробка пакета	Повторне надходження або повторна обробка вже отриманого повідомлення	Дублювання повідомлення або некоректний стан обробки	Контроль повторної обробки пакетів і фіксація факту обробки повідомлення
3	Компрометація локального сховища	Доступ до пристрою користувача або його локального сховища	Ризик доступу до значної частини локального контенту	Локальне шифроване зберігання основного контенту, захищене зберігання ключового матеріалу
4	Несанкціонована заміна активного пристрою	Спроба прив’язати новий пристрій або використати нечинний клієнтський контур	Паралельний або несанкціонований доступ до облікового запису	Правило одного активного пристрою, деактивація попереднього пристрою
5	Надмірний доступ серверного контуру	Участь сервера в маршрутизації, перевірці пакета та доставці	Ризик змішування службової ролі сервера з роллю сховища відкритого змісту	Обмеження сервера службовими діями, відсутність доступу до відкритого змісту повідомлень
6	Спотворення	Некоректне	Ризик	Політики

№	Ризик	Причина	Можливий наслідок	Механізм зменшення
	резервної копії	створення або відновлення резервної копії	некоректного відновлення стану	резервування, контроль службових операцій, аудит критичних подій
7	Обхід процедури відновлення доступу	Використання відновлення доступу для несанкціонованого входу	Отримання доступу поза логікою активного пристрою	Контроль службових процедур, аудит, обмеження доступу до відновлення
8	Відновлення глобально видаленого контенту з резервної копії	Наявність раніше створеної резервної копії після глобального видалення	Повернення контенту, який уже недоступний у поточному стані	Фіксація меж глобального видалення, політики резервування і резервного зберігання

Критичними для цієї моделі є локальне сховище, активний пристрій і резервні копії. Локальне зберігання зменшує залежність від сервера, але підвищує значення клієнтського середовища: у разі компрометації пристрою ризик стосується не одного повідомлення, а всього локального контенту. Серверний контур також потребує контролю, бо бере участь у маршрутизації, перевірці службового формату пакета, доставці та тимчасовому зберіганні зашифрованих недоставлених повідомлень, але не повинен перетворюватися на постійне сховище листування.

Активний пристрій є чинним клієнтським контуром доступу. Якщо пристрій замінюється, попередній має втратити активний статус. Глобальне видалення також має межі: воно може змінювати стан повідомлення в активному контурі, але не гарантує фізичного усунення всіх копій, які могли бути створені раніше в резервному контурі.

Обмеження моделі визначають, які сценарії входять до базового розгляду, а які не повинні включатися без окремого обґрунтування. Основні обмеження подано в табл. 1.4.

Таблиця 1.4

Основні обмеження запропонованої моделі

№	Обмеження	Зміст обмеження	Наслідок для моделі
1	Персональний формат один-на-один	Базова модель орієнтована на діалог між двома користувачами	Групові чати та складна багатостороння взаємодія не входять до основного розгляду
2	Один активний пристрій	Користувач працює через один чинний клієнтський пристрій	Під час зміни пристрою попередній пристрій має втратити активний статус
3	Службова роль сервера	Сервер виконує маршрутизацію, перевірку службового формату пакета та доставку	Сервер не є основним сховищем відкритого змісту листування
4	Тимчасове офлайн-зберігання	Зашифровані недоставлені повідомлення можуть зберігатися в серверному контурі до шести місяців	Офлайн-зберігання не змінює службову роль сервера
5	Первинний цикл доставки	Передбачено до трьох повторних спроб у межах 10 секунд	Доставка має обмежений початковий цикл
6	Обмеження розміру файла	Максимальний розмір файла становить 50 МБ	Модель не охоплює передавання більших файлів без додаткового обґрунтування
7	Кількість локально доступних чатів	У моделі задано до 500 локально доступних чатів	Локальне сховище розглядається в межах заданого обсягу
8	Допоміжний корпоративний контур	Корпоративний контур використовується для резервування, аудиту, адміністрування або відновлення доступу	Корпоративний контур не змінює базову модель персонального захищеного обміну

Ці обмеження не слід трактувати як недоліки системи. Вони зберігають фокус на контрольованій моделі захищеного десктопного месенджера, у якій основний контент зберігається локально, сервер виконує службову роль, а користувач працює через один активний пристрій.

Отже, аналіз ризиків і обмежень уточнює, які частини системи потребують найбільшого контролю, і створює основу для огляду існуючих рішень та порівняння їх із запропонованою моделлю.

1.4 Огляд існуючих рішень у сфері захищених месенджерів

Огляд існуючих рішень потрібний для уточнення місця запропонованої моделі серед наявних підходів до захищеного обміну повідомленнями. Це не маркетинговий аналіз конкурентів, а порівняння релевантних месенджерів за критеріями, важливими для теми дипломної роботи: захищений обмін, роль сервера, зберігання даних, десктопний формат, багатопристроєвість і службові або корпоративні можливості.

Для огляду обрано Signal, Telegram, WhatsApp, Element/Matrix і Threema. Порівняння виконується тільки за підтвердженими характеристиками і не містить тверджень про абсолютну захищеність або перевагу одного рішення над іншим.

Signal використовує end-to-end encryption для повідомлень і дзвінків. Сервіс не має доступу до змісту повідомлень і дзвінків, історія повідомлень зберігається на пристроях користувача, а зашифровані повідомлення можуть ставитися в чергу доставки для тимчасово недоступних пристроїв. Signal Desktop працює як linked device; офіційна довідка фіксує один мобільний пристрій і до п’яти linked Signal Desktop [1; 2].

Telegram розділяє cloud chats і Secret Chats. Cloud chats використовують сервер-клієнтське шифрування та хмарну синхронізацію, а Secret Chats використовують end-to-end encryption, не входять до Telegram Cloud і доступні

тільки на пристроях, де були створені. Telegram Business додає quick replies, automated messages, profile customization і chatbot integration [3; 4].

WhatsApp заявляє end-to-end encryption для персональних повідомлень і дзвінків. У режимі linked devices кожен linked device підключається незалежно, а особисті повідомлення, медіа й дзвінки залишаються захищеними end-to-end encryption. Також описані end-to-end encrypted backups і WhatsApp Business Platform як business messaging platform [5; 6; 7].

Element побудований на Matrix і позиціонується як Matrix-based end-to-end encrypted messenger and secure collaboration app. Matrix використовує федеративну архітектуру з homeserver-ами, які зберігають історію комунікації та інформацію акаунта своїх клієнтів і синхронізують історію з іншими homeserver-ами. Element також підтримує децентралізоване або self-hosted використання [8; 9].

Threema описує сервери як switch: повідомлення й дані пересилаються, але не зберігаються постійно. Водночас для асинхронної комунікації потрібне певне тимчасове зберігання. Контакти й групові чати зберігаються децентралізовано на пристроях користувачів. Для організацій доступні Threema Work і Threema OnPrem [10; 11].

Порівняння існуючих рішень за визначеними критеріями наведено в табл. 1.5.

Таблиця 1.5

Порівняння існуючих рішень у сфері захищених месенджерів

Рішення	Захищений обмін	Роль сервера	Зберігання даних	Службові можливості
Signal	End-to-end encryption для повідомлень і дзвінків	Сервер не має доступу до змісту; зашифровані повідомлення можуть ставитися в	Історія зберігається на пристроях; є secure backups і device transfers	Linked devices, secure backups, device transfers

		чергу доставки		
Telegram	End-to-end encryption у Secret Chats; cloud chats мають іншу модель	Cloud chats працюють через хмарну інфраструктуру; Secret Chats не входять до Telegram Cloud	Cloud chats синхронізуються через хмару; Secret Chats є device-specific	Telegram Business: quick replies, automated messages, chatbot integration
WhatsApp	End-to-end encryption для персональних повідомлень, медіа й дзвінків	Сервер бере участь у доставці, але не повинен мати доступу до відкритого змісту E2EE-повідомлень	Можливі end-to-end encrypted backups; linked device отримує E2EE-копію недавньої історії	WhatsApp Business Platform як business messaging platform
Element / Matrix	Element підтримує end-to-end encrypted messaging and calls	Федеративна архітектура з homeserver-ами	Homeserver зберігає історію комунікації та інформацію акаунта своїх клієнтів і синхронізує її з іншими homeserver-ами	Self-hosting, federated deployment, організаційні сценарії
Threema	End-to-end encryption; strong encryption на пристроях користувачів	Сервер виконує роль switch; повідомлення пересилаються, але не зберігаються постійно	Контакти й групи зберігаються децентралізовано; асинхронна комунікація передбачає тимчасове серверне зберігання	Threema Work і OnPrem, self-hosted deployment, адміністрування

Порівняння показує різні архітектурні підходи: Signal і WhatsApp демонструють end-to-end encrypted обмін із linked-device логікою, Telegram —

хмарний підхід з окремим режимом Secret Chats, Element/Matrix — федеративну модель із homeserver-ами та self-hosted сценаріями, Threema — мінімізацію постійного серверного зберігання й корпоративні варіанти використання.

Запропонована модель не дублює повністю жодне з цих рішень. Вона має власні межі: захищений десктопний застосунок, персональний формат один-на-один, локальне зберігання основного контенту, службову роль сервера, тимчасове зберігання лише зашифрованих недоставлених повідомлень і правило одного активного пристрою.

1.5 Постановка завдання

Після аналізу предметної області, вимог, ризиків, обмежень та існуючих рішень можна сформулювати постановку завдання для подальшого моделювання захищеного десктопного месенджера. У межах дипломної роботи потрібно не створити завершений промисловий месенджер, а сформувати концептуальну модель системи та розглянути окремі програмні фрагменти, які демонструють роботу ключових механізмів.

Розроблювана система належить до класу прикладних інформаційних систем для захищеного персонального обміну повідомленнями та файлами. Базова модель обмежується форматом один-на-один, спирається на локальне сховище користувача, службову роль серверного контуру й один активний пристрій.

У межах постановки завдання система має працювати з даними користувача, пристрою, діалогу, повідомлення, файла, квитанції та службового запиту. Це не довільний розширений набір сутностей, а постановочна формалізація даних, що випливають із вимог, ризиків і меж моделі. Вхідні дані, автоматизовану обробку та вихідні результати подано в табл. 1.6.

Таблиця 1.6

Вхідні дані, автоматизована обробка та вихідні результати системи

№	Вхідні дані	Автоматизована обробка	Вихідні результати

1	Дані користувача	Реєстрація, вхід до системи, перевірка доступу	Створений або підтверджений обліковий запис, стан авторизації
2	Дані активного пристрою	Прив’язка нового активного пристрою, перевірка чинного пристрою, деактивація попереднього пристрою	Чинний активний пристрій, втрата активного статусу попереднім пристроєм
3	Дані діалогу	Створення персонального діалогу один-на-один	Створений або оновлений діалог між двома користувачами
4	Текст повідомлення	Криптографічна обробка, формування службового пакета, маршрутизація через серверний службовий контур	Зашифрований пакет повідомлення, локально збережене повідомлення, стан доставки
5	Дані файла	Перевірка обмеження розміру, обробка файлового вкладення, збереження у локальному сховищі	Збережене файлове вкладення у локальному захищеному сховищі або відхилення файла, що не відповідає обмеженню
6	Дані квитанції	Фіксація службового стану доставки або прочитання	Оновлений стан повідомлення
7	Службові поля пакета	Перевірка службового формату, маршрутизація, контроль повторної обробки	Коректно оброблений або відхилений пакет
8	Запит глобального видалення	Зміна стану повідомлення в активному контурі з урахуванням меж резервного зберігання	Повідомлення, недоступне в активному контурі; межі впливу на резервні копії залишаються визначеними політиками

			резервування
9	Службовий запит корпоративного контуру	Резервування, аудит, адміністрування або відновлення доступу в межах допоміжного контуру	Результат службової процедури без деталізації формату службових записів у межах розділу 1

Таблиця 1.6 не є структурою класів або форматом бази даних. Детальна модель сутностей, зв’язків і процесів має розглядатися в розділі 2, а фрагменти програмної реалізації — у розділі 3.

Інформація, що зберігається в системі, розподіляється між локальним, серверним службовим і корпоративним допоміжним контурами. У локальному контурі зберігається основний контент: історія повідомлень, файлові вкладення, стани повідомлень і дані, потрібні для роботи десктопного застосунку. Серверний контур може тимчасово зберігати зашифровані недоставлені повідомлення до шести місяців. Корпоративний контур використовується для резервування, аудиту, адміністрування, політик резервування, адміністративних сповіщень, базової звітності або відновлення доступу.

Автоматизації підлягають реєстрація, вхід, прив’язка активного пристрою, створення діалогу один-на-один, обмін повідомленнями й файлами, формування квитанцій, локальне та глобальне видалення, обробка зашифрованих недоставлених повідомлень, контроль повторної обробки пакета, резервування, аудит і відновлення доступу. Глобальне видалення не можна трактувати як гарантоване фізичне усунення всіх попередньо створених резервних копій.

Очікуваний результат дипломної роботи — побудова концептуальної моделі захищеного десктопного месенджера та демонстрація окремих програмних механізмів. Завдання роботи й очікувані результати узагальнено в табл. 1.7.

Таблиця 1.7

Завдання дипломної роботи та очікуваний результат

№	Завдання	Очікуваний результат
1	Дослідити предметну область захищених десктопних месенджерів	Визначені основні інформаційні процеси, межі десктопної моделі, роль локального сховища, серверного службового контуру й активного пристрою
2	Сформувати вимоги до захищеного месенджера	Описані функціональні, безпекові, нефункціональні та службові вимоги до системи
3	Проаналізувати ризики й обмеження моделі	Визначені ризики локального сховища, активного пристрою, серверного службового контуру, резервних копій і глобального видалення
4	Оглянути існуючі рішення у сфері захищених месенджерів	Порівняні релевантні рішення та уточнено відмінність запропонованої моделі
5	Побудувати концептуальну модель системи	Визначені межі системи, ролі, сутності, зв’язки, основні процеси, життєві цикли та контури захищеності
6	Розглянути фрагменти програмного забезпечення	Описані програмні фрагменти для активного пристрою, захищеного обміну повідомленнями, роботи з файлами, локального захищеного сховища та глобального видалення

Отже, постановка завдання визначає межі подальшої роботи: потрібно побудувати контрольовану концептуальну модель захищеного десктопного застосунку з фрагментарною програмною демонстрацією ключових механізмів. Опорними елементами залишаються локальне зберігання основного контенту, службова роль сервера, один активний пристрій, персональний формат один-на-один, обробка повідомлень і файлів, резервування, аудит та контроль меж глобального видалення.

1.6 Висновки до розділу 1

У першому розділі виконано системний аналіз предметної області захищених десктопних месенджерів і сформовано постановку завдання для подальшого моделювання системи. Месенджер розглянуто як прикладну інформаційну систему для персонального обміну повідомленнями й файлами, роботи з обліковими записами, підтримки службових станів і захисту користувацького контенту.

Визначено межі запропонованої моделі: захищений десктопний застосунок, формат один-на-один, локальне зберігання основного контенту, службова роль серверного контуру та один активний пристрій. Сервер не є основним сховищем відкритого змісту листування; його роль обмежується маршрутизацією, перевіркою службового формату пакета, доставкою та тимчасовим зберіганням зашифрованих недоставлених повідомлень.

Сформовано функціональні, безпекові, нефункціональні та службові вимоги. Окремо визначено ризики локального сховища, активного пристрою, серверного службового контуру, резервних копій, відновлення доступу та глобального видалення. Глобальне видалення не подається як гарантоване фізичне усунення всіх попередньо створених резервних копій.

Огляд Signal, Telegram, WhatsApp, Element/Matrix і Threema показав різні підходи до захищеного обміну: end-to-end encryption, хмарну синхронізацію, linked devices, федеративну архітектуру, self-hosted сценарії, мінімізацію постійного серверного зберігання та корпоративні режими. Запропонована модель не дублює повністю жодне з цих рішень і має власні контрольовані межі.

У постановці завдання визначено, що подальша робота має бути спрямована на побудову концептуальної моделі захищеного десктопного месенджера та розгляд окремих програмних фрагментів. Наступний розділ має

визначити межі системи, ролі, сутності, зв’язки, інформаційні процеси, життєві цикли ключових об’єктів і контури захищеності.

РОЗДІЛ 2 КОНЦЕПТУАЛЬНЕ МОДЕЛЮВАННЯ ЗАХИЩЕНОГО ДЕСКТОПНОГО МЕСЕНДЖЕРА

Після визначення проблеми, мети, обмежень і загальної постановки задачі в розділі 1 необхідно перейти до концептуального моделювання майбутньої системи. У цьому розділі розглядається внутрішня логіка захищеного десктопного месенджера: межі системи, ролі учасників, модель даних, організація інформаційної бази, правила доступу, процес захищеного обміну, життєві цикли основних об’єктів і модель захищеності. Такий опис потрібний для того, щоб узгодити аналітичні висновки першого розділу з подальшим поданням програмних фрагментів у розділі 3.

2.1 Межі системи, ролі та доступ

Захищений десктопний месенджер у межах цієї роботи моделюється як система персонального обміну повідомленнями і файлами. Базова модель спирається на діалоги один-на-один, локальне зберігання основного контенту, службову роль сервера та доступ користувача через один активний пристрій. Саме ці положення визначають межі системи й відокремлюють її від універсального месенджера з ширшою функціональністю.

До базової моделі входять клієнтський десктопний застосунок, активний пристрій користувача, локальне захищене сховище, пов’язаний із пристроєм ключовий контур, механізми обміну повідомленнями і файлами, а також службовий серверний контур. Серверна частина потрібна для перевірки службового формату пакета, маршрутизації, службового забезпечення доставки, фіксації даних для аудиту критичних подій і тимчасового зберігання зашифрованих недоставлених повідомлень.

Користувач у цій моделі працює через один активний пристрій. Такий пристрій є чинним клієнтським контуром доступу до локального контенту й службових дій. Якщо користувач змінює активний пристрій, попередній пристрій має втратити активний статус. Це обмеження потрібне для того, щоб доступ до локального контенту й ключового матеріалу не розподілявся між невизначеною кількістю клієнтських середовищ.

Серверний контур не розглядається як основне сховище відкритого змісту повідомлень. Його роль є службовою: він бере участь у маршрутизації, перевірці службових ознак пакета, доставці, обробці офлайн-сценаріїв і тимчасовому зберіганні зашифрованих недоставлених повідомлень. Така межа дозволяє відокремити прикладну логіку користувача від службової інфраструктури передавання.

Корпоративний контур має допоміжний характер. Він може використовуватися для резервування, аудиту, адміністрування або відновлення доступу, але не змінює основний фокус базової моделі: локальне зберігання основного контенту, службову роль сервера, контроль активного пристрою та персональний формат обміну повідомленнями.

Таблиця 2.1

Ролі та межі доступу в системі

Роль / контур	Призначення	Межі доступу
Користувач	Працює з повідомленнями, файлами, квитанціями, локальним і глобальним видаленням.	Доступ виконується через один активний пристрій.
Активний пристрій	Є чинним клієнтським контуром доступу користувача.	Після зміни пристрою попередній пристрій втрачає активний статус.
Клієнтський застосунок	Виконує прикладну логіку обміну, локального зберігання та роботи з файлами.	Не замінює серверний або корпоративний контур.
Серверний службовий	Маршрутизація,	Не є основним сховищем

Роль / контур	Призначення	Межі доступу
контур	доставка, перевірка службового формату, тимчасове зберігання зашифрованих недоставлених пакетів.	відкритого змісту повідомлень.
Корпоративний контур	Резервування, аудит, адміністрування, відновлення доступу.	Має допоміжний характер і не змінює базову модель one-to-one.

У таблиці 2.1 подано ролі й контури, які використовуються в концептуальній моделі. Їх опис обмежується рівнем проєктної логіки, без переходу до програмних класів або прав доступу на рівні коду.

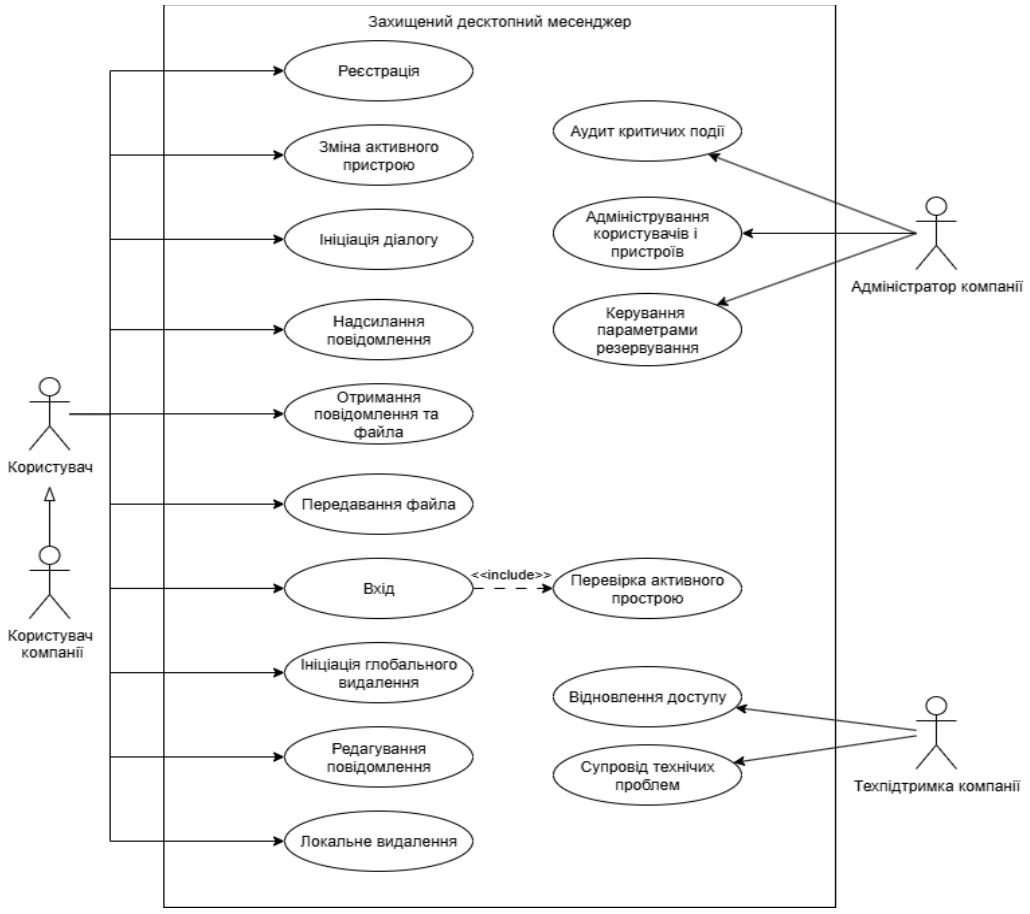

Рис. 2.1: діаграма варіантів використання «Межі системи та доступ користувачів»

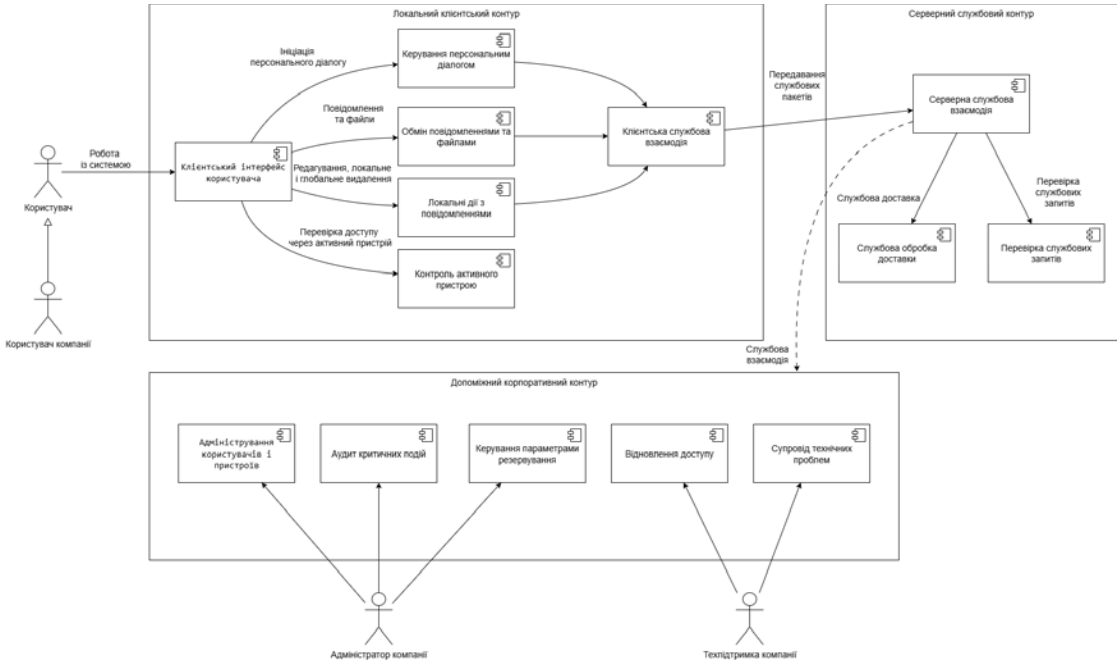

Рис. 2.2: діаграма компонентів «Контури доступу захищеного десктопного месенджера»

Отже, у підрозділі визначено межі системи та ролі основних учасників моделі. Це створює основу для подальшого опису сутностей, зв’язків і логічної організації даних.

2.2 Концептуальна та логічна модель даних

Після визначення меж системи, ролей і доступу потрібно описати логічну структуру даних, на які спирається захищений десктопний месенджер. У цьому підрозділі дані розглядаються не як фізичні таблиці бази даних, а як концептуальні сутності, що відображають основні об’єкти системи та зв’язки між ними. Такий підхід дозволяє показати, які елементи формують базову прикладну логіку месенджера, які підтримують доступ через активний пристрій, а які належать до службового або корпоративного рівня.

Основу моделі становлять користувач, пристрій, локальний чат, повідомлення, файл і квитанція. Користувач є центральною сутністю, оскільки

саме з ним пов’язані обліковий контур, активний пристрій, локальні чати, повідомлення та службові дії. Пристрій у цій моделі не є другорядним технічним елементом, оскільки доступ до системи пов’язується з одним активним пристроєм. Через це він відображає не лише місце запуску десктопного застосунку, а й частину довіреного клієнтського середовища.

Локальний чат описує персональну взаємодію один-на-один. Він не моделюється як груповий простір, тому що групові чати не входять до базової моделі. Повідомлення є основним об’єктом обміну і пов’язується зі станами доставки, службовими ознаками, квитанціями, можливістю локального або глобального видалення та файловими вкладеннями. Файл у моделі не розглядається як простий додаток до повідомлення: з ним пов’язані обмеження розміру, локальне зберігання, передавання та захист контенту.

Квитанція потрібна для фіксації службових результатів доставки або прочитання. Запит глобального видалення моделюється окремим службовим об’єктом, оскільки глобальне видалення не збігається з локальним видаленням. Окремо враховуються резервні копії та аудит, які належать до корпоративного або службового рівня і не змінюють базову модель персонального обміну.

Таблиця 2.2

Сутності логічної моделі даних

Сутність	Призначення	Роль у моделі
User	Користувач системи.	Пов’язаний з обліковим контуром, активним пристроєм, чатами й службовими діями.
Device	Пристрій користувача.	Фіксує активний клієнтський контур доступу.
LocalChat	Локальний чат один-на-один.	Відображає персональну взаємодію між двома користувачами.
Message	Повідомлення.	Основний об’єкт обміну зі станами, службовими ознаками й можливістю

Сутність	Призначення	Роль у моделі
		видалення.
File	Файлове вкладення.	Пов’язане з повідомленням, обмеженням розміру, локальним зберіганням і захистом контенту.
Receipt	Квитанція доставки або прочитання.	Фіксує службовий результат обробки повідомлення.
GlobalDeleteRequest	Запит глобального видалення.	Окремий службовий об’єкт для ініціювання глобального видалення.
Backup	Резервна копія.	Використовується в корпоративному контурі як зашифрований службовий об’єкт.
AuditRecord	Службовий запис аудиту.	Фіксує критичні події без розкриття змісту повідомлень.

У таблиці 2.2 узагальнено основні сутності логічної моделі. Вона не переходить до фізичної схеми бази даних, а визначає змістові об’єкти, з якими працює система.

Таблиця 2.3

Основні зв’язки між сутностями логічної моделі

Зв’язок	Між якими сутностями	Значення для моделі
Користувач має активний пристрій	User — Device	Обмежує доступ одним чинним клієнтським контуром.
Користувач бере участь у локальному чаті	User — LocalChat	Підтримує формат one-to-one.
Чат містить повідомлення	LocalChat — Message	Формує локальну історію обміну.
Повідомлення може мати файл	Message — File	Пов’язує текстовий і файловий контент.
Повідомлення має квитанції	Message — Receipt	Фіксує доставку або прочитання.
Повідомлення може мати запит глобального видалення	Message — GlobalDeleteRequest	Відокремлює глобальне видалення від локального видалення.

Зв’язок	Між якими сутностями	Значення для моделі
Дані можуть потрапляти в резервний контур	Message / File — Backup	Пояснює межу резервування без гарантії фізичного очищення всіх копій.
Службові дії фіксуються аудитом	Device / Message / GlobalDeleteRequest — AuditRecord	Підтримує контроль критичних подій у службовому або корпоративному контурі.

Зв’язки між сутностями показують, що повідомлення в моделі не зводиться до окремого текстового запису. Воно пов’язане з користувачем, активним пристроєм, локальним чатом, файлами, квитанціями, запитами глобального видалення, резервуванням і аудитом. Саме така структура потрібна для подальшого моделювання процесів і станів.

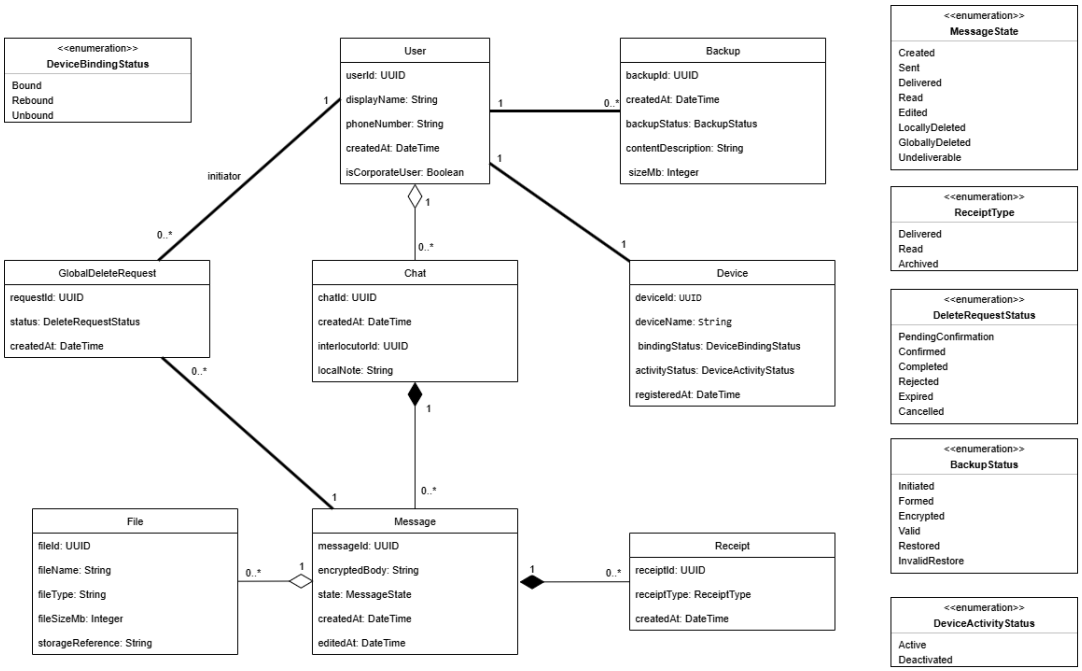

Рис. 2.3: діаграма класів «Концептуальна та логічна модель даних захищеного десктопного месенджера»

У результаті концептуальна та логічна модель даних визначає основу, на якій надалі будуються процеси роботи месенджера. Вона показує, які сутності беруть участь в обміні повідомленнями і файлами, як доступ пов’язується з

активним пристроєм, які службові об’єкти потрібні для глобального видалення, резервування та аудиту, а також як корпоративний контур доповнює базову модель.

2.3 Організація інформаційної бази та правила доступу до даних

Після визначення сутностей і зв’язків логічної моделі потрібно описати, де зберігаються дані та які контури мають до них доступ. Організація інформаційної бази в цій роботі розглядається на концептуальному рівні: вона показує розподіл даних між локальним клієнтським середовищем, серверним службовим контуром і корпоративним контуром, але не переходить до опису SQL-операцій або програмного коду.

Локальна база SQLite використовується як сховище прикладних і службових даних клієнтського застосунку. До неї належать локальні чати, повідомлення, службові ознаки станів, квитанції, локальні відомості про файли та дані, потрібні для роботи клієнта. Водночас ключовий матеріал не розглядається як звичайний запис SQLite, оскільки він належить до окремого ключового контуру, пов’язаного із системно захищеним сховищем операційної системи.

Файловий контент у моделі відокремлюється від службових записів. Це потрібно для того, щоб не змішувати метадані повідомлення, його стан і сам файловий вміст. Файл пов’язується з повідомленням, але його обробка має власні обмеження, зокрема максимальний розмір вкладення та потребу в захищеному зберіганні.

Серверний службовий контур використовується для маршрутизації, перевірки службового формату пакета, доставки та тимчасового зберігання зашифрованих недоставлених пакетів. Він не є основним сховищем відкритого змісту повідомлень. У випадку офлайн-сценарію сервер може тимчасово

зберігати зашифрований пакет до доставки або до завершення визначеного строку зберігання.

Корпоративний контур пов’язаний із резервуванням, аудитом, адмініструванням і відновленням доступу. Він має допоміжний характер і не змінює базову модель персонального обміну один-на-один. Резервні копії в цій моделі не треба трактувати як відкриті копії повідомлень; вони мають зберігатися як зашифровані службові об’єкти.

Таблиця 2.4

Розподіл даних між контурами зберігання

Тип даних	Основне місце зберігання	Хто має доступ	Особливість
Локальні чати й повідомлення	Клієнтський застосунок / SQLite	Користувач через активний пристрій	Основний зміст пов’язаний із пристроєм користувача.
Файловий контент	Локальне файлове сховище клієнта	Користувач через клієнтський застосунок	Файл пов’язаний із повідомленням і має окремі обмеження.
Квитанції та службові стани	Локальна база та службовий контур	Клієнтський застосунок і службові механізми	Потрібні для контролю доставки й прочитання.
Зашифрований недоставлений пакет	Серверний службовий контур	Сервер як службовий посередник	Зберігається без відкритого змісту.
Резервні копії	Корпоративний контур	Адміністративні та службові механізми	Розглядаються як зашифровані службові об’єкти.
Ключовий матеріал	Системно захищене сховище ОС	Активний пристрій і захисний механізм ОС	Не зберігається як звичайний запис SQLite.
Службові записи аудиту	Корпоративний або службовий контур	Службові механізми контролю	Не містять відкритого змісту повідомлень.

У таблиці 2.4 розмежовано дані за контурами зберігання, щоб показати, які об’єкти залишаються локальними, а які належать до серверного або корпоративного контуру.

Таблиця 2.5

Правила доступу до даних за контурами

Контур	До чого має доступ	До чого не має доступу
Локальний довірений контур	Локальні повідомлення, файли, квитанції, службові стани активного користувача.	Дані інших користувачів поза межами локального контексту.
Серверний службовий контур	Службовий формат пакета, маршрут доставки, зашифровані недоставлені пакети.	Відкритий зміст повідомлень і файлових даних.
Корпоративний контур	Зашифровані резервні копії, аудит, адміністративні службові дані.	Відкритий зміст повідомлень без відповідних локальних умов доступу.
Системно захищене сховище ОС	Захищений локальний ключовий матеріал.	Прикладні записи SQLite як звичайні дані.

Таблиця 2.5 доповнює це розмежування правилами доступу до відповідних груп даних.

Таблиця 2.6

Концептуальний сценарій відновлення після збою

Ситуація	Що використовується	Очікуваний результат	Межа
Збій клієнтського застосунку	Локальна база та локальні службові стани	Відновлення локального стану після повторного запуску	Без опису програмного алгоритму відновлення.
Недоставлене повідомлення	Серверний службовий контур	Повторна доставка або завершення зберігання за встановленою межею	Без доступу сервера до відкритого змісту.
Пошкодження або	Корпоративний	Можливість	Склад, формат і

втрата локального середовища	контур і резервні копії	службового відновлення в межах наявних резервних даних	частота резервних копій не деталізуються.
Зміна активного пристрою	Ключовий контур і службове підтвердження зміни	Попередній пристрій втрачає активний статус	Повний життєвий цикл ключів не деталізується.

Сценарій відновлення після збою подано на концептуальному рівні. Таблиця 2.6 не визначає програмний алгоритм, а лише показує, які контури можуть брати участь у відновленні працездатності або службового стану системи.

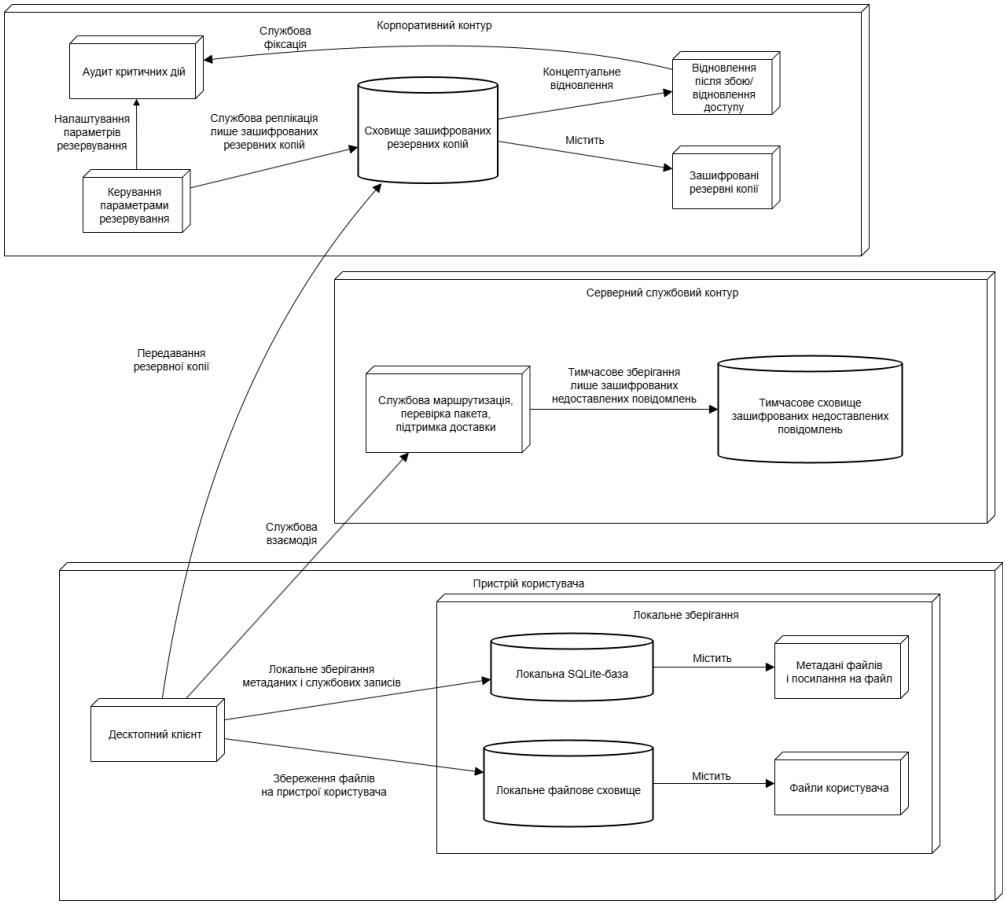

Рис 2.4: діаграма розгортання «Організація інформаційної бази та контури зберігання»

Отже, організація інформаційної бази ґрунтується на розмежуванні локального зберігання, серверної службової обробки та корпоративного резервування. Після розмежування даних і правил доступу можна перейти до опису того, як повідомлення та файли проходять через систему під час захищеного обміну.

2.4 Моделювання процесу захищеного обміну повідомленнями та файлами

Основні процеси месенджера показують, як визначені раніше сутності переходять у взаємодію між користувачами, пристроями, локальним сховищем і серверним службовим контуром. У межах цієї моделі до таких процесів належать ініціація діалогу один-на-один, підготовка повідомлення, надсилання й отримання повідомлення, передавання файла, формування квитанцій, локальне збереження контенту та обробка негативних умов.

Першим процесом є ініціація діалогу один-на-один. У цій моделі діалог не створюється як довільна групова взаємодія, оскільки базовий формат системи обмежений персональним обміном між двома сторонами. Після створення локального чату користувач може підготувати повідомлення або файлове вкладення для передавання.

Підготовка повідомлення відбувається на стороні клієнтського застосунку. Перед передаванням система має врахувати активний пристрій користувача, доступність необхідних локальних умов обробки, допустимість файлового вкладення та готовність службового контуру до приймання пакета. Якщо умови не виконані, надсилання має бути відхилене або відкладене відповідно до негативного сценарію.

У разі успішної підготовки повідомлення або файл передається через серверний службовий контур. Сервер перевіряє службовий формат пакета, визначає маршрут доставки й передає пакет одержувачу. Якщо адресат недоступний, серверний контур може тимчасово зберігати зашифрований

недоставлений пакет у межах визначеного строку. При цьому сервер не отримує відкритого змісту повідомлення або файла.

На стороні одержувача пакет проходить службову перевірку, після чого повідомлення може бути оброблене клієнтським застосунком, збережене в локальному сховищі та підтверджене квитанцією. Квитанції доставки або прочитання не замінюють повідомлення, а фіксують службовий результат його проходження через основні етапи обміну.

Таблиця 2.7

Етапи процесу захищеного обміну повідомленнями та файлами

Етап	Вхідні дані	Дія системи	Результат
Ініціація діалогу	Користувач і адресат	Створення або відкриття локального чату один-на-один	Підготовлено контекст обміну.
Підготовка повідомлення	Текст повідомлення або файл	Перевірка активного пристрою, умов обробки та службових обмежень	Повідомлення готове до передавання або відхилене.
Передавання пакета	Підготовлений захищений пакет	Передавання через серверний службовий контур	Пакет спрямовано до адресата або тимчасово збережено.
Офлайн-доставка	Зашифрований недоставлений пакет	Тимчасове зберігання та повторна доставка	Пакет доставлено або завершено обробку за межею зберігання.
Отримання повідомлення	Пакет на стороні одержувача	Службова перевірка й локальна обробка	Повідомлення доступне в локальному клієнтському середовищі.
Формування квитанції	Результат доставки або прочитання	Створення службового підтвердження	Стан повідомлення оновлено.
Обробка негативної умови	Помилка формату, недоступність	Відхилення, повторна спроба	Система переходить до

Етап	Вхідні дані	Дія системи	Результат
	адресата, порушення обмежень	або тимчасове зберігання	визначеного негативного сценарію.

У таблиці 2.7 процес обміну подано як послідовність етапів — від підготовки повідомлення до доставки, квитанцій і обробки негативних умов. Опис залишається концептуальним і не переходить до коду або конкретного мережевого протоколу.

У результаті процесна модель показує, як повідомлення або файл проходить шлях від підготовки до доставки, локального збереження та підтвердження квитанцією. Негативні умови не виносяться за межі моделі, оскільки вони впливають на подальші стани повідомлення та службових запитів.

2.5 Моделювання життєвих циклів повідомлення, запиту глобального видалення та негативних сценаріїв

Після опису процесу захищеного обміну потрібно визначити, як змінюються стани основних об’єктів системи. У цьому підрозділі розглядаються концептуальні життєві цикли повідомлення та запиту глобального видалення, а також негативні сценарії, які можуть виникати під час передавання, доставки, зберігання або службової обробки.

Стани повідомлення подаються як концептуальні. Вони потрібні для моделювання поведінки системи, але не є остаточними програмними enum-станами. Це розмежування важливе, оскільки розділ 2 описує модель, а не програмну реалізацію. У реалізації частина станів може бути деталізована або згрупована інакше, але модель має показати логіку переходів між основними етапами обробки повідомлення.

Таблиця 2.8

Концептуальні стани повідомлення

Стан повідомлення	Умова переходу	Зміст стану
Створене локально	Користувач підготував повідомлення	Повідомлення ще не передане до службового контуру.
Підготовлене до передавання	Перевірено локальні умови й активний пристрій	Повідомлення готове до формування захищеного пакета.
Передане до серверного контуру	Пакет передано серверу	Сервер виконує службову перевірку та маршрутизацію.
Тимчасово збережене як недоставлене	Адресат недоступний	Зашифрований пакет очікує доставки в межах строку зберігання.
Доставлене	Пакет отримано клієнтом адресата	Повідомлення пройшло етап доставки.
Прочитане	Одержувач відкрив повідомлення	Формується або оновлюється службова квитанція.
Локально видалене	Користувач виконав локальне видалення	Повідомлення видалене лише в локальному контексті.
Очікує глобального видалення	Створено запит глобального видалення	Повідомлення пов’язане з окремою службовою процедурою.
Обробка завершена	Досягнуто кінцевого стану доставки, видалення або відхилення	Повідомлення більше не потребує активної службової обробки.

Таблиця 2.8 описує концептуальні стани повідомлення. Локальне видалення і глобальне видалення не змішуються, оскільки глобальне видалення моделюється через окремий службовий запит.

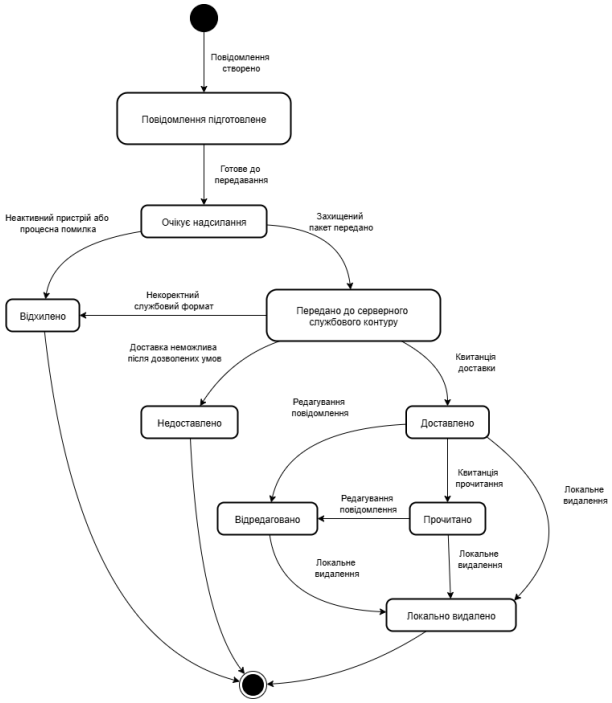

Рис. 2.6: діаграма станів «Концептуальний життєвий цикл повідомлення»

Запит глобального видалення відокремлюється від самого повідомлення, оскільки він є службовим об’єктом, що змінює стан обробки, але не гарантує фізичного очищення всіх попередньо створених резервних копій. Таке обмеження потрібне для коректного розмежування прикладної дії користувача, службової обробки та резервного контуру.

Таблиця 2.9

Стани запиту глобального видалення

Стан запиту	Умова переходу	Результат
Створено	Користувач ініціював глобальне видалення	Сформовано службовий запит.
Очікує перевірки	Запит передано до службового контуру	Виконується перевірка службових ознак.

Стан запиту	Умова переходу	Результат
Підтверджено до обробки	Перевірка успішна	Запит може змінити стан пов’язаного повідомлення.
Відхилено	Запит некоректний або не підтверджений	Глобальне видалення не виконується.
Виконано в межах моделі	Обробка завершена	Стан повідомлення оновлено відповідно до запиту.
Завершено з обмеженням	Є резервні або службові межі	Не гарантується фізичне очищення всіх попередніх резервних копій.

Таблиця 2.9 показує, що запит глобального видалення має власний життєвий цикл. Завершення такого запиту не означає, що всі резервні копії або попередні службові стани фізично усунені.

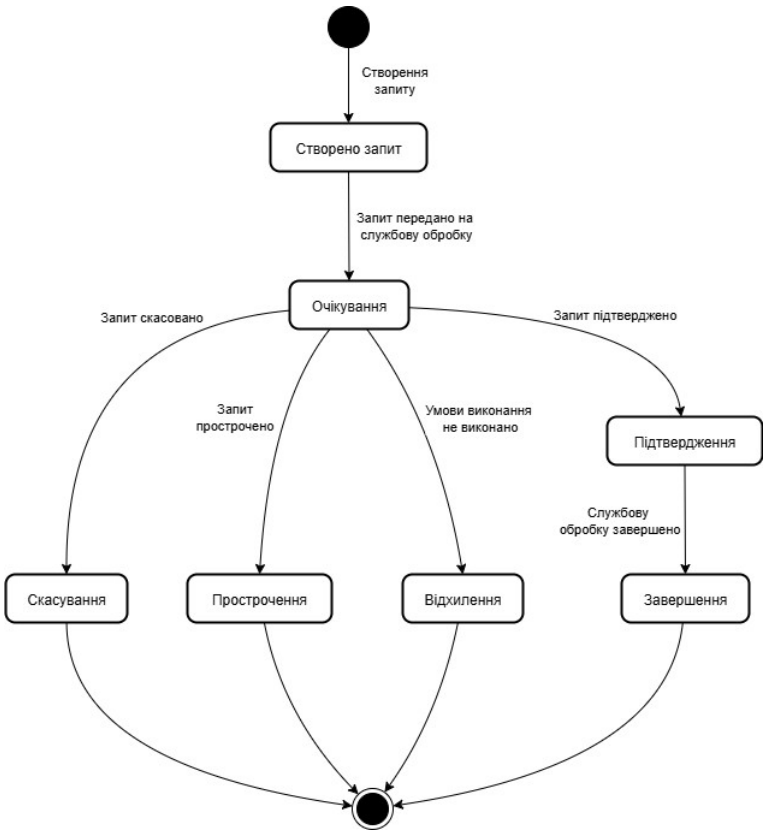

Рис. 2.7: діаграма станів «Життєвий цикл запиту глобального видалення»

Таблиця 2.10

Негативні сценарії та реакція системи

Негативний сценарій	Умова виникнення	Реакція системи	Межа
Недоступність адресата	Одержувач перебуває офлайн	Тимчасове зберігання зашифрованого недоставленого пакета або повторна доставка	Не довше визначеної межі зберігання.
Перевищення розміру файла	Файл перевищує допустиме обмеження	Відхилення передавання	Без автоматичного обходу обмеження.
Неактивний пристрій	Користувач працює не через активний пристрій	Відмова в обробці службової дії	Попередній пристрій не має чинного статусу.
Некоректний службовий пакет	Порушено службовий формат або ознаки пакета	Відхилення обробки	Без розкриття відкритого змісту.
Повторна обробка пакета	Пакет уже був оброблений або має ознаки повтору	Відхилення повторної обробки	Без створення дубліката повідомлення.
Некоректний запит глобального видалення	Запит не проходить службову перевірку	Відхилення запиту	Стан повідомлення не змінюється.
Обмеження резервного контуру	Повідомлення або службові дані вже потрапили в резервну копію	Фіксація межі глобального видалення	Не гарантується фізичне очищення всіх копій.

Негативні сценарії потрібні для того, щоб модель не описувала лише успішний обмін. Вони показують, як система має поводитися в ситуаціях, коли порушено умови доставки, активного пристрою, службового формату, розміру файла або глобального видалення.

Отже, підрозділ визначає життєві цикли повідомлення, запиту глобального видалення та негативні сценарії. Після визначення станів і

негативних сценаріїв можна окремо розглянути, які об’єкти потребують захисту та які межі має модель захищеності.

2.6 Модель захищеності системи та криптографічні інструменти

Після визначення меж системи, логічної моделі даних, організації інформаційної бази, процесу обміну та життєвих циклів основних об’єктів потрібно окремо описати модель захищеності. У цьому підрозділі розглядаються криптографічні інструменти, їхня роль у системі, контури захисту, основні загрози та межі застосування обраних рішень. Опис має концептуальний характер: він не задає формат захищеного пакета, не визначає параметри ключів, не описує nonce/IV, API, SQL-операції або програмний код.

Модель захищеності системи спирається на те, що відкритий зміст повідомлень і файлових даних не має бути доступним серверному службовому контуру. Сервер бере участь у службовій передачі, зберіганні недоставлених зашифрованих пакетів і синхронізації службових станів, але не розглядається як контур обробки відкритого змісту. Резервування в межах моделі належить до корпоративного контуру. Тому для власної моделі не використовується термін end-to-end encryption: у роботі достатньо зафіксувати, що контент передається й зберігається у зашифрованому вигляді, а серверний контур не має доступу до відкритого змісту повідомлень.

У моделі захищеності доцільно розрізняти три контури. Локальний довірений контур охоплює пристрій користувача, локальний контент, локальний ключовий матеріал і засоби його захисту. Серверний службовий контур відповідає за службову передачу, перевірку службових ознак, доставку та тимчасове зберігання зашифрованих недоставлених пакетів без доступу до відкритого змісту. Корпоративний контур охоплює зашифровані резервні копії, службову реплікацію, аудит критичних подій і адміністративні функції без розкриття відкритого змісту повідомлень.

У підрозділі використовується поняття «захищений пакет». Воно означає концептуальний контейнер, який поєднує зашифрований контент і службові ознаки, необхідні для обробки процесу обміну. У межах цього підрозділу не визначаються поля пакета, порядок їх серіалізації, формат підпису, структура службових метаданих або правила формування nonce/IV. Таке обмеження потрібне для того, щоб не перетворити концептуальну модель на непідтверджену реалізаційну специфікацію.

Основними криптографічними інструментами моделі є AES-GCM, SHA-256, RSA-PSS/SHA-256 та DPAPI / ProtectedData. AES-GCM використовується в моделі для шифрування контенту повідомлень і файлових даних [12]. SHA-256 використовується для контрольних хешів контенту або файлових даних без опису полів захищеного пакета [13]. RSA-PSS/SHA-256 застосовується для підпису визначених службових дій, зокрема пакета повідомлення, запиту глобального видалення, квитанції та зміни активного пристрою [14]. DPAPI / ProtectedData використовується як Windows-орієнтований механізм захисту локального ключового матеріалу.

У межах цієї роботи застосунок розглядається як Windows-орієнтований. Через це DPAPI / ProtectedData розглядається саме як механізм, доступний у Windows-середовищі. Інші операційні системи, альтернативні сховища ключів і кросплатформні механізми захисту локального ключового матеріалу в цій роботі не аналізуються.

Таблиця 2.11

Криптографічні інструменти та їх призначення

Інструмент	Призначення в моделі	Що захищає	Межа застосування
AES-GCM	Шифрування контенту	Текст повідомлення, файловий контент або вміст захищеного пакета	Без деталізації параметрів ключів, nonce/IV і формату пакета
SHA-256	Формування	Контент або	Без опису

Інструмент	Призначення в моделі	Що захищає	Межа застосування
	контрольних хешів	файлові дані	службових полів пакета
RSA-PSS/SHA-256	Підпис службових дій	Пакет повідомлення, запит глобального видалення, квитанція, зміна активного пристрою	Без зазначення розміру ключів і параметрів RSA-PSS
DPAPI / ProtectedData	Захист локального ключового матеріалу	Локальний ключовий матеріал на пристрої користувача	Windows-орієнтований механізм; інші ОС не розглядаються

У таблиці 2.11 криптографічні інструменти подано через їхню роль у моделі: шифрування контенту, контрольні хеші, підпис службових дій і захист локального ключового матеріалу. Розміри ключів, nonce/IV, поля пакета, параметри підпису або програмні бібліотеки свідомо не подаються, оскільки такі дані не були визначені як частина концептуальної моделі.

Ключовий матеріал не зберігається як звичайний запис SQLite. SQLite у попередніх підрозділах розглядається як локальна база для прикладних і службових даних клієнтського застосунку, але не як звичайне сховище ключового матеріалу [15]. Локальний ключовий матеріал пов’язується із системно захищеним сховищем операційної системи через DPAPI / ProtectedData. Зміна активного пристрою деактивує попередній ключовий контур, але повний життєвий цикл ключів, зокрема генерація, ротація, відкликання, відновлення та резервування ключів, у цьому підрозділі не деталізується.

Окремого розмежування потребують об’єкти захисту та загрози, які стосуються цих об’єктів. У системі захищаються не тільки повідомлення, а й файловий контент, локальний ключовий матеріал, службові дії, резервні копії, недоставлені зашифровані пакети та службові записи аудиту. Для кожного з цих

об’єктів застосовується окрема роль криптографічного або службового механізму, що не повинна змішуватися з процесами зберігання або життєвими циклами, описаними в попередніх підрозділах.

Таблиця 2.12

Об’єкти захисту, загрози та засоби захисту

Об’єкт захисту	Основна загроза	Засіб або механізм зменшення ризику	Межа застосування
Контент повідомлення	Розкриття відкритого змісту під час передавання або зберігання	AES-GCM	Не описується формат захищеного пакета
Файлові дані	Розкриття або неконтрольована зміна файлового контенту	AES-GCM, SHA-256	Не описується структура файла або службових метаданих
Локальний ключовий матеріал	Неконтрольований доступ до ключового матеріалу на пристрої користувача	DPAPI / ProtectedData	Windows-орієнтований захист; ключовий матеріал не зберігається як звичайний запис SQLite
Пакет повідомлення	Перехоплення, підміна або неавторизована зміна службового пакета	RSA-PSS/SHA-256	Не описуються поля пакета й параметри підпису
Запит глобального видалення	Несанкціонована або некоректна службова дія щодо повідомлення	RSA-PSS/SHA-256	Не гарантує фізичного очищення всіх резервних копій
Квитанція доставки або прочитання	Підміна службового підтвердження	RSA-PSS/SHA-256	Не описується формат квитанції
Зміна активного пристрою	Несанкціонована зміна ключового контуру	RSA-PSS/ SHA-256, DPAPI / ProtectedData	Повний key lifecycle не деталізується
Недоставлений зашифрований	Розкриття змісту під час	AES-GCM	Серверний контур не отримує

пакет	тимчасового зберігання		відкритого змісту
Зашифрована резервна копія	Розкриття резервних даних або помилкове трактування резервної копії як відкритого сховища	AES-GCM, службова реплікація лише зашифрованих копій	Склад, формат, частота й механізм очищення не деталізуються
Службові записи аудиту	Втрата фіксації критичних подій або неможливість простежити службову дію	Службова фіксація критичних подій у корпоративному контурі	Не описується формат журналу, API, SQL або повна модель аудиту

У таблиці 2.12 для кожного об’єкта захисту визначено основну загрозу, засіб зменшення ризику та межу застосування відповідного інструмента або службового механізму.

Аудит у цій моделі розглядається як службова фіксація критичних подій у корпоративному контурі. Він не перетворюється на окрему реалізаційну модель журналювання. У межах підрозділу не визначаються структура журналу, формат запису, SQL-таблиці, API або механізм збереження аудиту. Достатньо зафіксувати, що аудит підтримує контроль службових дій, але не відкриває зміст повідомлень і не змінює криптографічну модель.

Резервні копії в цій моделі розглядаються тільки як зашифровані службові об’єкти. Службова реплікація стосується лише зашифрованих резервних копій. Склад резервної копії, її внутрішній формат, частота створення, порядок зберігання та механізм очищення не деталізуються. Це узгоджується з обмеженням попередніх підрозділів: глобальне видалення не гарантує фізичного очищення всіх раніше створених резервних копій.

Модель захищеності також має чіткі межі. Вона не описує повний криптографічний протокол, схему обміну ключами, PKI, TLS, API, SQL, поля пакета або програмний код. Таке обмеження не є пропуском змісту, а фіксує рівень деталізації розділу 2. У межах розділу 2 потрібно показати проєктні

рішення, їхню роль і взаємозв’язок із попередніми підрозділами. Реалізаційні фрагменти, які демонструють окремі частини програмної логіки, належать до розділу 3.

Таблиця 2.13

Межі криптографічної моделі

Елемент, який не деталізується	Причина обмеження	Як це враховано в підрозділі
Поля захищеного пакета	Формат пакета не визначено як частину концептуальної моделі	Використовується тільки поняття концептуального контейнера
nonce/IV для AES-GCM	Параметри реалізації не підтверджені	AES-GCM описано лише як засіб шифрування контенту
Розміри ключів	Не визначено в наданих матеріалах	Не вказуються конкретні значення
Параметри RSA-PSS	Не визначено в наданих матеріалах	RSA-PSS/SHA-256 описано тільки як підпис службових дій
Повний key lifecycle	Генерація, ротація, відкликання й відновлення ключів не деталізовані	Фіксуються лише межі захисту локального ключового матеріалу
PKI або сертифікати	Інфраструктура сертифікатів не описана	Не вводиться новий механізм довіри
TLS або мережевий протокол	Мережевий протокол не є предметом 2.6	Не додається окрема транспортна модель
API	Належить до реалізації	Не описується в концептуальній моделі
SQL-операції	Належать до фізичної реалізації	Не змішуються з моделлю захищеності
Код	Належить до розділу 3	2.6 задає модельні рішення, а не програмну реалізацію
Кросплатформний захист ключів	Застосунок орієнтований на Windows	Інші ОС не розглядаються
Очищення резервних копій	Механізм очищення не підтверджений	Фіксується тільки зашифрований характер резервних копій
Формат службових записів аудиту	Модель журналювання не деталізується	Аудит описано лише як службову фіксацію

Елемент, який не деталізується	Причина обмеження	Як це враховано в підрозділі
		критичних подій
Повне доведення властивостей алгоритмів	Потребує офіційних джерел і не є метою підрозділу	У тексті вказано призначення інструментів без реалізаційних параметрів

У таблиці 2.13 зафіксовано межі криптографічної моделі, щоб відокремити концептуальні рішення від реалізаційних параметрів.

Діаграма для підрозділу 2.6 не є обов'язковою. Основний зміст цього підрозділу подано через таблиці, оскільки вони точніше фіксують призначення інструментів, об'єкти захисту, загрози та межі моделі. Якщо діаграма буде додана, вона має залишатися на концептуальному рівні й не повинна розкривати формат пакета або реалізаційні параметри криптографічних алгоритмів.

Підрозділ 2.6 узгоджує криптографічні інструменти з об'єктами захисту, контурами системи та службовими обмеженнями. Завдяки цьому модель захищеності не відривається від попередніх підрозділів і не переходить до непідтверджених реалізаційних деталей. AES-GCM відповідає за шифрування контенту, SHA-256 — за контрольні хеші контенту або файлових даних, RSA-PSS/SHA-256 — за підпис службових дій, а DPAPI / ProtectedData — за захист локального ключового матеріалу в Windows-середовищі. Серверний контур не має доступу до відкритого змісту повідомлень, ключовий матеріал не зберігається як звичайний запис SQLite, аудит фіксує критичні службові події без розкриття змісту, а резервні копії розглядаються лише як зашифровані службові об'єкти. Це завершує концептуальне пояснення захищеності розділу 2 і створює основу для демонстрації окремих програмних фрагментів у розділі 3.

2.7 Висновок до розділу 2

У другому розділі сформовано концептуальну модель захищеного десктопного месенджера. На відміну від розділу 1, де було визначено проблему, мету роботи, обмеження та загальну постановку задачі, у цьому розділі описано

внутрішню логіку майбутньої системи: її межі, ролі, дані, контури зберігання, процеси обміну, життєві цикли ключових об’єктів і модель захищеності.

UML-діаграми, передбачені для розділу, доповнюють текстове моделювання та формалізують межі системи, модель даних, процеси, життєві цикли й контури захищеності. Вони не замінюють текстове пояснення, а використовуються для наочного подання тих зв’язків і переходів, які були описані в підрозділах 2.1–2.6.

Спочатку було визначено межі системи, ролі учасників і правила доступу. Основною моделлю взаємодії залишено персональний обмін повідомленнями один-на-один. Для доступу користувача враховано правило одного активного пристрою, а серверний контур розглянуто як службовий елемент, що не є основним сховищем відкритого змісту повідомлень. Таке розмежування дозволило відокремити локальний довірений контур користувача, серверний службовий контур і корпоративний контур. Корпоративний контур доповнює базову модель резервуванням, аудитом і відновленням доступу, але не змінює її основну логіку.

Далі було побудовано концептуальну та логічну модель даних. У ній визначено основні сутності, зв’язки між ними та службові об’єкти, потрібні для обміну повідомленнями, роботи з файлами, квитанціями, глобальним видаленням і контролем активного пристрою. У моделі відокремлено об’єкти, пов’язані з користувачем, пристроєм, локальним чатом, повідомленням, файлами, квитанціями, глобальним видаленням, резервуванням і аудитом. Логічна модель не замінює фізичну реалізацію бази даних, але задає основу для подальшого програмного проєктування.

Окремо розглянуто організацію інформаційної бази та правила доступу до даних. Локальна база SQLite використовується для прикладних і службових даних клієнтського застосунку, але не розглядається як звичайне сховище ключового матеріалу. Файловий контент, серверний службовий контур, резервні копії та корпоративний контур мають різні ролі, тому в розділі було

розмежовано локальне зберігання, тимчасову службову обробку та резервування.

Процес захищеного обміну повідомленнями та файлами описано як послідовність дій від підготовки повідомлення до передавання, доставки, отримання квитанцій і обробки негативних умов. У межах цього процесу збережено обмеження щодо активного пристрою, розміру файлового вкладення, повторних спроб доставки та тимчасового зберігання зашифрованих недоставлених пакетів. Серверний контур у цій моделі виконує службову роль і не отримує відкритого змісту повідомлень.

Після процесної моделі було визначено життєві цикли повідомлення та запиту глобального видалення. Стани повідомлення подано як концептуальні, а не як остаточні програмні enum-стани. Локальне видалення відокремлено від глобального видалення, оскільки глобальне видалення моделюється окремим службовим запитом. Також зафіксовано, що завершення такого запиту не означає гарантованого фізичного очищення всіх раніше створених резервних копій.

У підрозділі про модель захищеності визначено криптографічні інструменти та межі їх застосування. AES-GCM використовується в моделі для шифрування контенту, SHA-256 — для контрольних хешів контенту або файлових даних, RSA-PSS/SHA-256 — для підпису визначених службових дій, а DPAPI / ProtectedData — для захисту локального ключового матеріалу в Windows-середовищі. Водночас розділ не переходить до повної криптографічної специфікації: у ньому не визначаються поля захищеного пакета, nonce/IV, розміри ключів, API, SQL-операції або програмний код.

Отже, у розділі 2 сформовано цілісну концептуальну основу захищеного десктопного месенджера. Вона поєднує межі системи, модель даних, організацію інформаційної бази, процес обміну, життєві цикли об’єктів і модель захищеності. Отримані рішення не є повною промисловою реалізацією, але

вони задають структурну й технічну основу для розділу 3, у якому мають бути розглянуті окремі програмні фрагменти реалізації.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНИХ ФРАГМЕНТІВ СИСТЕМИ

3.1 Організаційна структура програмного забезпечення та вибір інструментарію

У третьому розділі розглядається практична частина роботи, пов’язана з реалізацією окремих програмних фрагментів системи. Розділ не подає повну архітектуру завершеного програмного продукту, а зосереджується на фрагментах, які демонструють основну логіку роботи системи.

Реалізація програмних фрагментів виконувалася мовою C#. Як середовище розробки використовувалася Visual Studio 2022. Програмні фрагменти орієнтовані на платформу .NET 8, операційну систему Windows 11 і тип застосунку WinForms. Це середовище використовується для демонстраційної реалізації окремих частин системи без переходу до опису повного промислового розгортання.

Для збереження меж розділу програмну реалізацію подано як групи реалізаційних фрагментів, а не як повний набір модулів готового месенджера. Такий підхід дає змогу відокремити фактично описані частини від непідтверджених елементів: серверної маршрутизації, бази даних, адміністрування, резервного копіювання або повної інфраструктури готового продукту.

Таблиця 3.1

Реалізаційні групи програмних фрагментів

Реалізаційна група	**Призначення**	**Межі розкриття**
Прив’язка пристрою	Опис логіки зв’язування пристрою з користувачем	Розглядається фрагмент обробки запиту прив’язки без серверної інфраструктури та без повної транзакційної моделі
Обмін повідомленнями	Опис формування та	Розглядаються програмні дії

Реалізаційна група	Призначення	Межі розкриття
	отримання захищеного пакета повідомлення	створення й обробки пакета без мережевого транспорту та серверної доставки
Локальне збереження файлових даних	Опис збереження і читання файлового вмісту в межах локального фрагмента	Фізичний формат локального сховища не розкривається
Демонстраційна криптографічна обробка	Пояснення службової обробки даних у реалізаційних фрагментах	Не подається як промислова криптографічна інфраструктура і не містить тверджень про доведену криптографічну стійкість
Демонстраційна перевірка	Узагальнення перевірки поведінки визначених сценаріїв	Не подається як повне тестування, фактичний протокол запуску або оцінка продуктивності

Таблиця 3.1 фіксує ті реалізаційні напрями, які використовуються для подальшого опису в розділі. Вона не є схемою повної архітектури системи й не замінює концептуальну модель, подану в розділі 2. Її роль полягає в тому, щоб показати межі подальшого розкриття програмних фрагментів.

3.2 Фрагмент прив’язки пристрою

Фрагмент прив’язки пристрою розглядається як один із важливих реалізаційних фрагментів програмної системи. Він визначає логічний зв’язок між користувачем і пристроєм, з якого надалі може виконуватися робота із системою. У цьому підрозділі описується не повна серверна процедура реєстрації пристрою, а окремий сценарій обробки запиту прив’язки.

Основним методом фрагмента є BindDevice. Він приймає запит зі службовими даними пристрою, зокрема відкритим ключем і атестаційним токеном. PublicKey і AttestationToken розглядаються тільки як поля запиту [17]. У межах підрозділу не додається окрема інфраструктура атестації, серверна перевірка токена або повна транзакційна модель.

Логіка фрагмента передбачає перевірку вхідних даних, формування запису про прив’язку та повернення результату. Якщо обов’язкові дані відсутні або не відповідають очікуваній структурі, прив’язка завершується неуспішно.

Якщо запит придатний для обробки, формується результат успішної прив’язки пристрою.

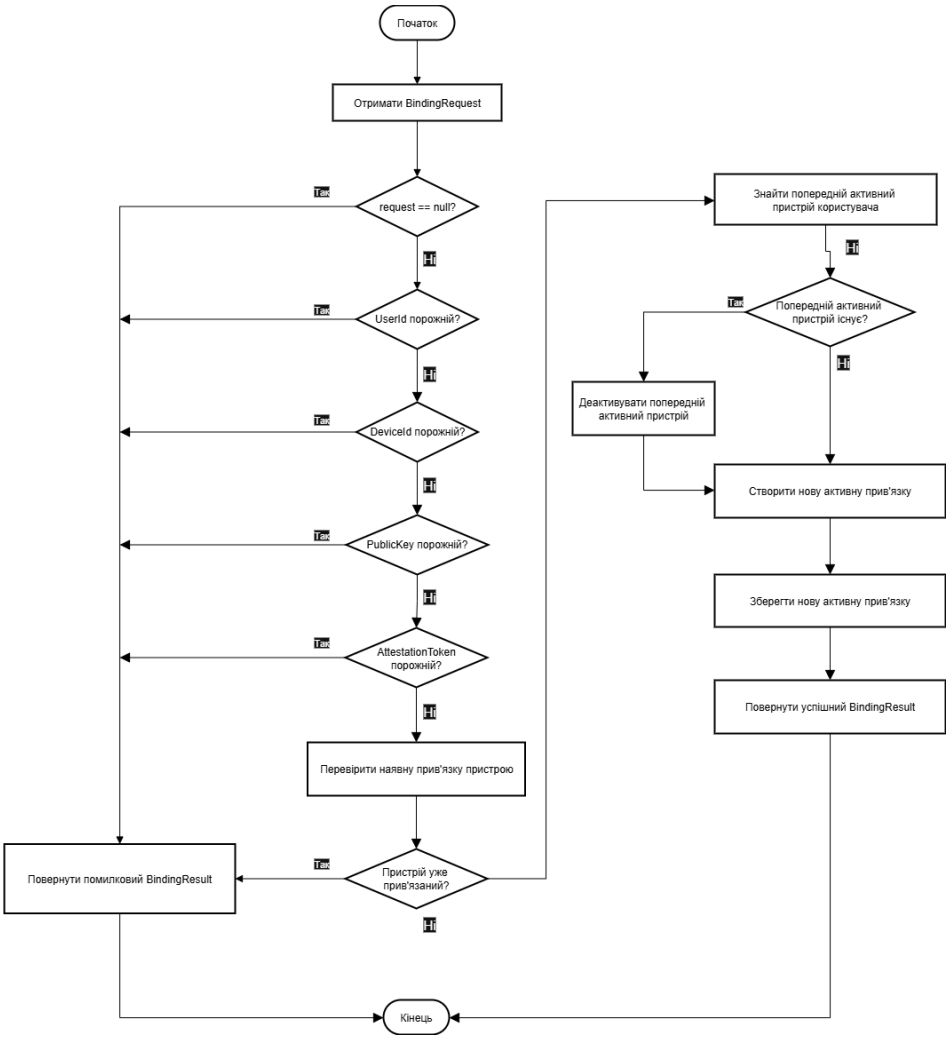

Рис. 3.1 UML activity-діаграма фрагмента прив’язки пристрою

Activity-діаграма показує послідовність обробки запиту прив’язки пристрою: отримання даних, перевірку умов і формування результату. Вона не описує серверну інфраструктуру або зовнішній механізм атестації.

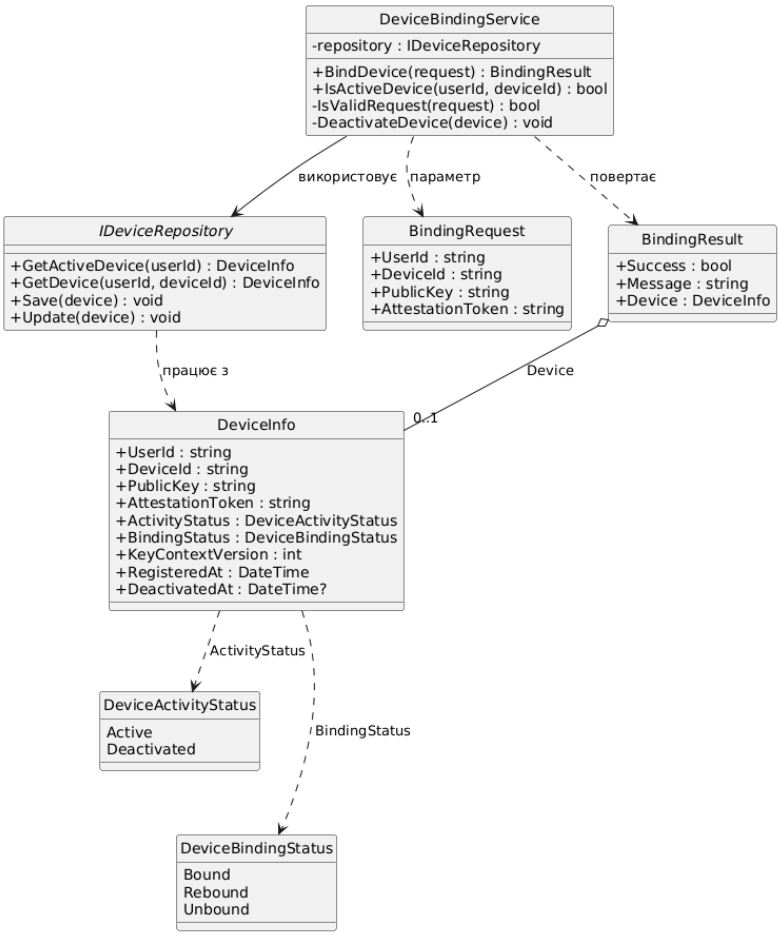

Рис. 3.2 UML class-діаграма фрагмента прив’язки пристрою

Class-діаграма показує основні класи, запит, результат і залежності, потрібні для пояснення фрагмента BindDevice. Вона не розширюється до повної моделі всіх компонентів системи.

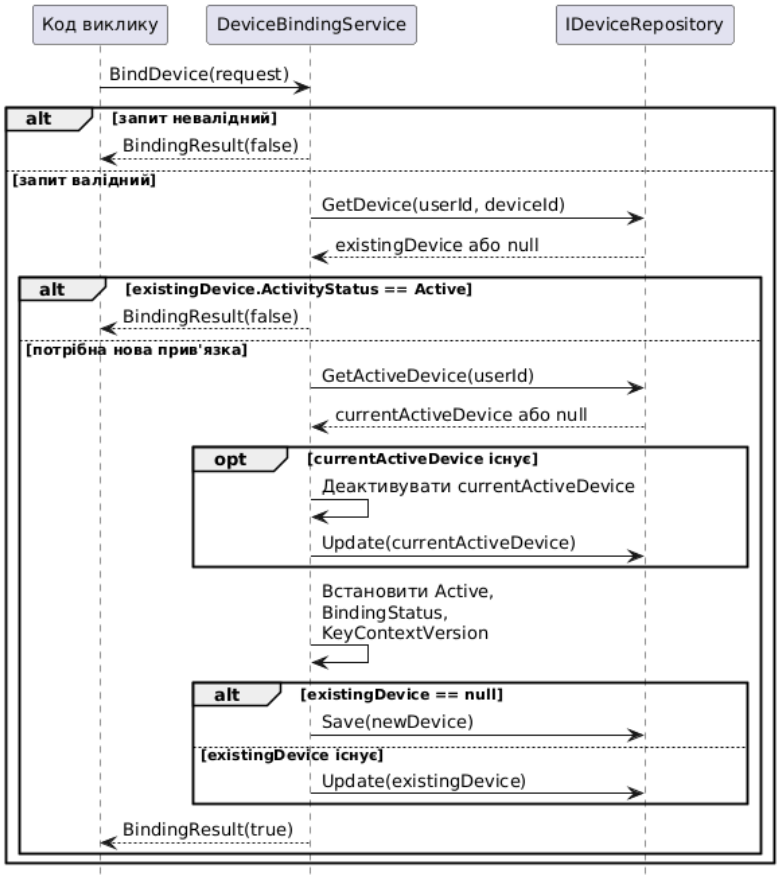

Рис. 3.3 UML sequence-діаграма фрагмента прив’язки пристрою

Sequence-діаграма уточнює порядок взаємодії об’єктів під час виконання фрагмента. Її призначення — показати внутрішню послідовність викликів, а не зовнішню мережеву взаємодію.

Лістинг 3.1 Фрагмент методу BindDevice

```
public DeviceBindingResult BindDevice(DeviceBindingRequest request)
{
    if (request == null)
        return DeviceBindingResult.Fail("Запит прив'язки відсутній.");
    if (string.IsNullOrWhiteSpace(request.UserId))
        return DeviceBindingResult.Fail("Не вказано користувача.");
    if (string.IsNullOrWhiteSpace(request.DeviceId))
        return DeviceBindingResult.Fail("Не вказано пристрій.");
    if (string.IsNullOrWhiteSpace(request.PublicKey))
        return DeviceBindingResult.Fail("Відкритий ключ відсутній.");
    if (string.IsNullOrWhiteSpace(request.AttestationToken))
        return DeviceBindingResult.Fail("Атестаційний токен відсутній.");
    if (_repository.Exists(request.UserId, request.DeviceId))
        return DeviceBindingResult.Fail("Пристрій уже прив'язано.");
    var binding = DeviceBinding.Create(
        request.UserId,
        request.DeviceId,
        request.PublicKey,
```

```
            request.AttestationToken);
        _repository.Save(binding);
        return DeviceBindingResult.Success(binding.BindingId);
    }
```

У лістингу подано компактний фрагмент методу BindDevice. Код демонструє перевірку запиту, обробку UserId, DeviceId, PublicKey, AttestationToken, перевірку повторної прив’язки, створення DeviceBinding і повернення результату. Повний код фрагмента винесено в додаток А.

Таблиця 3.2

Перевірка фрагмента прив’язки пристрою

№	**Сценарій перевірки**	**Вхідні дані / умова**	**Очікуваний результат**	**Статус підтвердження**
1	Коректний запит прив’язки пристрою	Заповнені UserId, DeviceId, PublicKey, AttestationToken; запис про таку прив’язку відсутній	Формується DeviceBinding, запис зберігається через репозиторій, повертається успішний результат із BindingId	Відповідає логіці фрагмента коду
2	Запит прив’язки відсутній	request == null	Повертається неуспішний результат із повідомленням про відсутній запит	Відповідає логіці фрагмента коду
3	Не вказано користувача	UserId порожній або складається з пробілів	Повертається неуспішний результат із повідомленням про відсутнього користувача	Відповідає логіці фрагмента коду
4	Не вказано пристрій	DeviceId порожній або складається з пробілів	Повертається неуспішний результат із повідомленням про відсутній пристрій	Відповідає логіці фрагмента коду
5	Відсутній відкритий ключ	PublicKey порожній або складається з пробілів	Повертається неуспішний результат із повідомленням про відсутній відкритий ключ	Відповідає логіці фрагмента коду
6	Відсутній атестаційний токен	AttestationToken порожній або складається з	Повертається неуспішний результат із	Відповідає логіці фрагмента коду

№	Сценарій перевірки	Вхідні дані / умова	Очікуваний результат	Статус підтвердження
		пробілів	повідомленням про відсутній атестаційний токен	
7	Повторна прив’язка того самого пристрою	Репозиторій уже містить пару UserId / DeviceId	Повертається неуспішний результат із повідомленням про вже прив’язаний пристрій	Відповідає логіці фрагмента коду

Таблиця 3.2 узагальнює демонстраційну перевірку фрагмента прив’язки пристрою. Вона не є фактичним протоколом запуску тестів і не підтверджує повне покриття всіх можливих сценаріїв.

3.3 Фрагмент захищеного обміну повідомленнями

Фрагмент захищеного обміну повідомленнями описує програмну логіку створення та обробки пакета повідомлення. У межах підрозділу не розглядаються серверна доставка, мережевий транспорт, endpoint-и або маршрутизація. Основна увага приділяється діям, які виконуються під час формування пакета та його подальшої обробки.

Основними методами фрагмента є CreatePacket і ReceivePacket. Метод CreatePacket формує захищений пакет повідомлення. Метод ReceivePacket обробляє отриманий пакет, перевіряє його службові ознаки та формує результат. Для пояснення структури обміну використовується модель EncryptedMessagePacket, а результати обробки подаються через MessageProcessingResult і MessageReceipt.

CreatePacket відповідає за підготовку повідомлення до подальшої обробки в межах програмної логіки фрагмента. ReceivePacket працює вже зі сформованим пакетом: перевіряє його, відновлює вміст і повертає результат обробки. PlainText розглядається тільки як оперативне значення після

розшифрування. Фізичний формат локального збереження повідомлень у цьому підрозділі не розкривається.

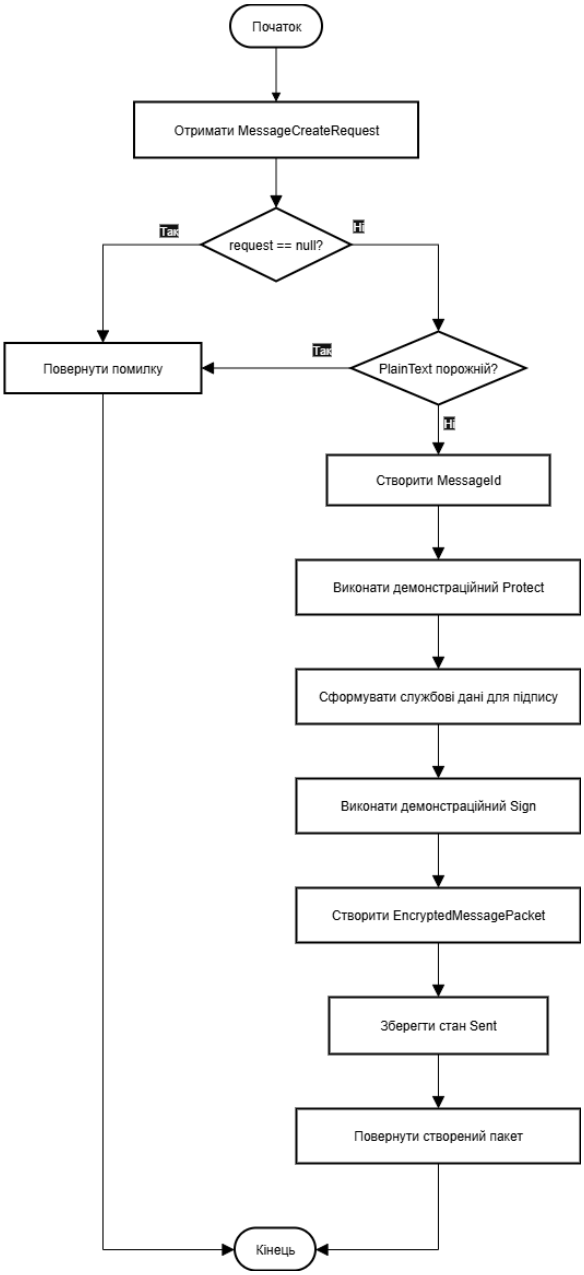

Рис. 3.4 UML activity-діаграма створення пакета повідомлення

Activity-діаграма для CreatePacket показує послідовність формування пакета повідомлення. Вона не описує мережеву доставку або серверну обробку.

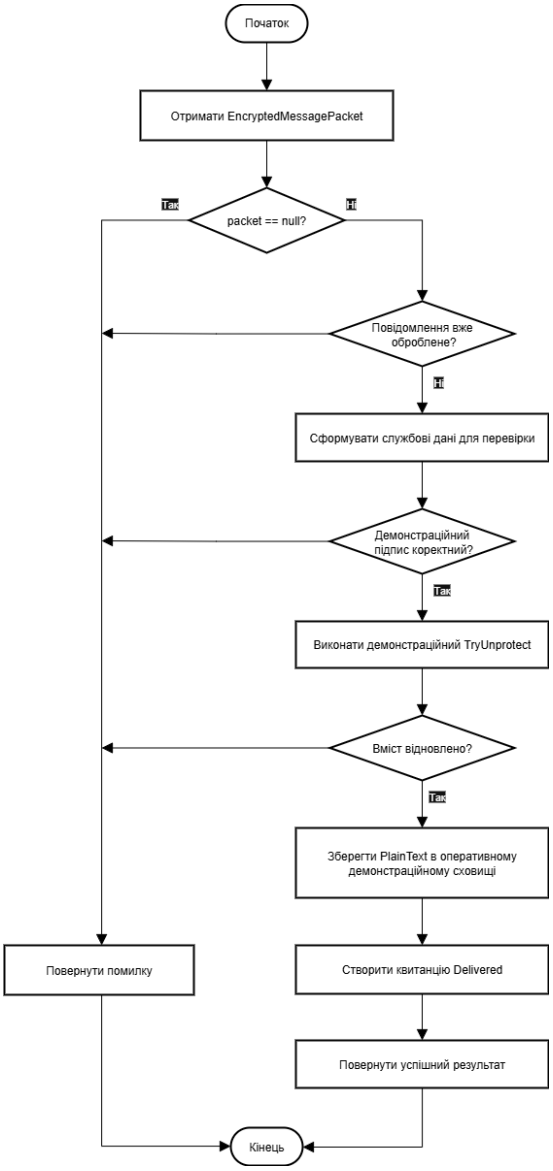

Рис. 3.5 UML activity-діаграма отримання пакета повідомлення

Activity-діаграма для ReceivePacket показує логіку обробки отриманого пакета: перевірку службових даних, відновлення вмісту та формування результату.

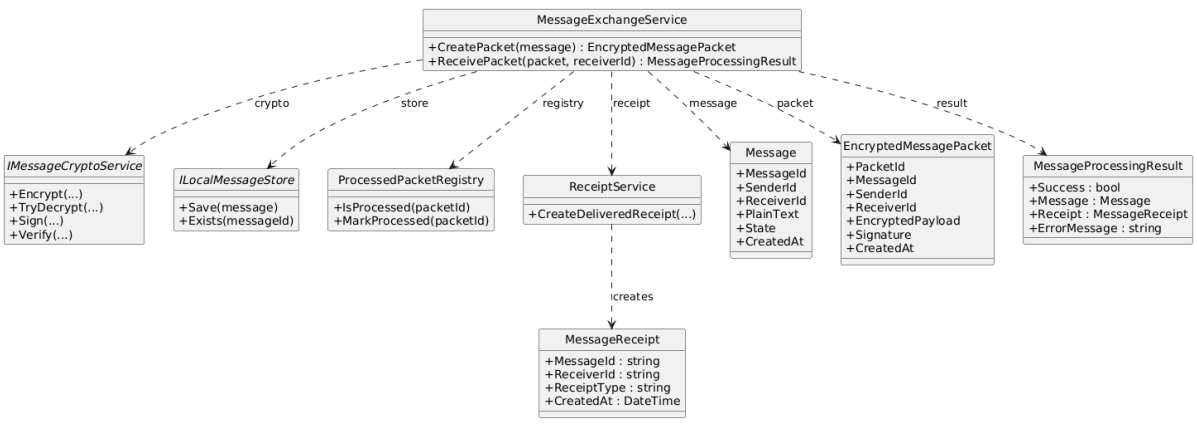

Рис. 3.6 UML class-діаграма фрагмента захищеного обміну повідомленнями

Class-діаграма відображає основний сервіс, пов’язані інтерфейси, моделі та результати. Центральною моделлю фрагмента є EncryptedMessagePacket.

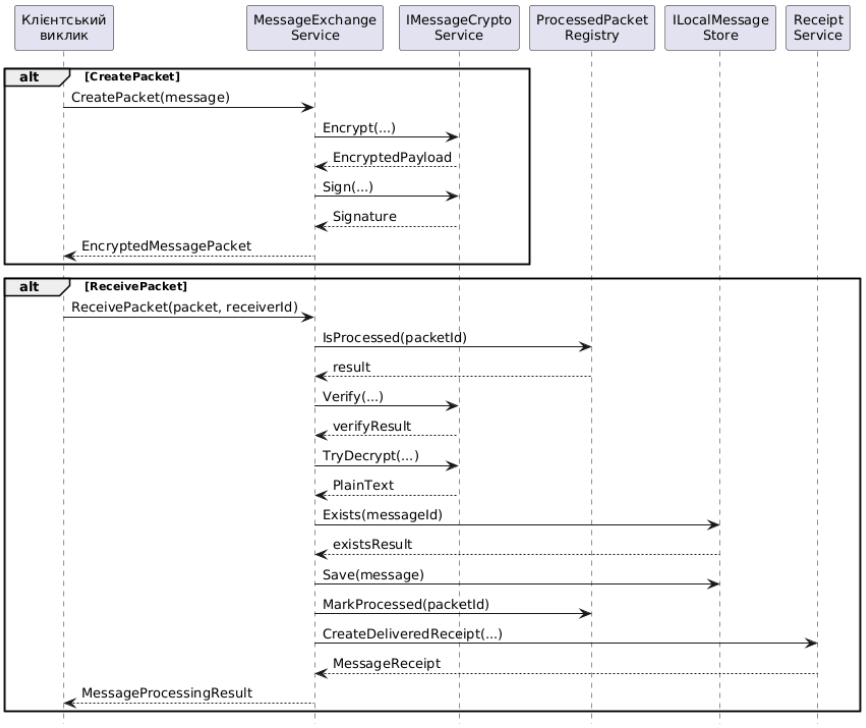

Рис. 3.7 UML sequence-діаграма фрагмента захищеного обміну повідомленнями

Sequence-діаграма показує порядок взаємодії об’єктів у двох сценаріях: створення пакета та отримання пакета. Вона не є описом повного мережевого протоколу.

Таблиця 3.3

Структура EncryptedMessagePacket

Поле	**Тип**	**Призначення у фрагменті**	**Обмеження використання**
MessageId	string	Службовий ідентифікатор пакета повідомлення	Використовується для розрізнення пакетів; не описує серверний ідентифікатор
SenderDeviceId	string	Ідентифікатор пристрою-відправника	Не розкриває повну модель користувача

Поле	Тип	Призначення у фрагменті	Обмеження використання
		в межах пакета	або маршрутизацію
ReceiverDeviceId	string	Ідентифікатор пристрою-отримувача в межах пакета	Не описує мережеву доставку повідомлення
CipherText	string	Захищене подання текстового вмісту повідомлення у демонстраційному фрагменті	Не є описом промислового криптографічного формату
Signature	string	Службове значення для перевірки даних пакета у фрагменті	Не подається як промислова цифрова підписна інфраструктура
CreatedAt	DateTime	Час створення пакета, який входить до службових даних	Не використовується як мережевий часовий протокол або серверний журнал

Таблиця 3.3 фіксує склад пакета повідомлення на рівні моделі, яка використовується у фрагменті. Вона не додає мережевий формат передавання, серверний протокол або фізичне локальне сховище.

Лістинг 3.2 Фрагменти методів CreatePacket і ReceivePacket

```
public EncryptedMessagePacket CreatePacket(MessageCreateRequest request)
{
    if (request == null)
        throw new ArgumentNullException(nameof(request));
    if (string.IsNullOrWhiteSpace(request.PlainText))
        throw new InvalidOperationException("Текст повідомлення відсутній.");
    var packet = new EncryptedMessagePacket
    {
        MessageId = Guid.NewGuid().ToString("N"),
        SenderDeviceId = request.SenderDeviceId,
        ReceiverDeviceId = request.ReceiverDeviceId,
        CipherText = _crypto.Protect(request.PlainText),
        CreatedAt = DateTime.UtcNow
    };
    packet.Signature = _crypto.Sign(BuildSignatureData(packet));
    _store.SaveState(packet.MessageId, MessageState.Sent);
    return packet;
}
public MessageProcessingResult ReceivePacket(EncryptedMessagePacket packet)
{
    if (packet == null)
        return Failure("Пакет повідомлення відсутній.");
    if (_store.Exists(packet.MessageId))
        return Failure("Повідомлення з таким ідентифікатором уже оброблено.");
    if (!_crypto.Verify(BuildSignatureData(packet), packet.Signature))
        return Failure("Службові дані пакета не пройшли перевірку.");
    if (!_crypto.TryUnprotect(packet.CipherText, out var plainText))
        return Failure("Вміст пакета не вдалося відновити.");
    _store.SaveReceived(packet.MessageId, plainText);
    var receipt = _receiptService.CreateDeliveredReceipt(packet.MessageId);
```

```
        return MessageProcessingResult.Success(plainText, receipt);
    }
    private static string BuildSignatureData(EncryptedMessagePacket packet)
    {
        return string.Join("|", packet.MessageId, packet.SenderDeviceId,
            packet.ReceiverDeviceId, packet.CipherText, packet.CreatedAt.Ticks);
    }
    private static MessageProcessingResult Failure(string reason)
    {
        return MessageProcessingResult.Failure(reason);
    }
```

У лістингу подано компактні фрагменти, потрібні для пояснення логіки створення та обробки пакета. CreatePacket формує EncryptedMessagePacket, ReceivePacket перевіряє пакет і повертає результат обробки. BuildSignatureData і Failure залишаються допоміжними частинами, потрібними для розуміння основної логіки. Повний код фрагмента винесено в додаток Б.

Демонстраційна криптографічна обробка використовується лише для пояснення механізму захищеного обміну в межах програмного фрагмента. Вона не подається як промислова криптографічна інфраструктура, не вводить конкретні алгоритми і не підтверджує криптографічну стійкість системи.

Таблиця 3.4

Демонстраційна перевірка фрагмента захищеного обміну повідомленнями

№	**Сценарій перевірки**	**Вхідні дані / умова**	**Очікуваний результат**	**Статус підтвердження**
1	Створення пакета з коректного запиту	MessageCreateRequest містить SenderDeviceId, ReceiverDeviceId і PlainText	Повертається EncryptedMessagePacket із MessageId, CipherText, CreatedAt і Signature; стан повідомлення зберігається як Sent	Відповідає логіці фрагмента коду
2	Створення пакета без запиту	request == null	Генерується ArgumentNullException	Відповідає логіці фрагмента коду
3	Створення пакета без тексту повідомлення	PlainText порожній або складається з пробілів	Генерується InvalidOperationException із повідомленням про відсутній текст	Відповідає логіці фрагмента коду
4	Отримання відсутнього пакета	packet == null	Повертається неуспішний MessageProcessin	Відповідає логіці фрагмента коду

№	Сценарій перевірки	Вхідні дані / умова	Очікуваний результат	Статус підтвердження
			gResult	
5	Повторна обробка повідомлення	Сховище вже містить MessageId отриманого пакета	Повертається неуспішний результат із повідомленням про вже оброблене повідомлення	Відповідає логіці фрагмента коду
6	Пакет із некоректними службовими даними	Verify повертає false для BuildSignatureData(packet) і Signature	Повертається неуспішний результат перевірки службових даних	Відповідає логіці фрагмента коду
7	Неможливість відновити вміст пакета	TryUnprotect повертає false	Повертається неуспішний результат із повідомленням про неможливість відновлення вмісту	Відповідає логіці фрагмента коду
8	Коректне отримання пакета	Пакет не дублюється, службові дані проходять перевірку, CipherText відновлюється	Зберігається отриманий PlainText, формується MessageReceipt зі станом Delivered, повертається успішний результат	Відповідає логіці фрагмента коду

Таблиця 3.4 подає демонстраційні сценарії перевірки фрагмента. Вона не є фактичним протоколом запуску тестів і не містить метрик продуктивності або повного тестового покриття.

3.4 Фрагмент локального збереження файлових даних

Фрагмент локального збереження файлових даних розглядається як окрема частина програмної логіки, що доповнює обмін повідомленнями. У межах підрозділу не розкривається фізичний формат локального сховища, конкретний шлях збереження, структура каталогу, база даних або формат файлів на диску.

Основним сервісом фрагмента є LocalFileStorageService. Для запису файлового вмісту використовується метод SaveFile(string fileName, byte[] content), а для читання — TryReadFile(string fileId, out byte[] content). Дані про файл подаються через LocalFileRecord, який містить FileId, FileName, ProtectedContent, ContentHash і CreatedAt.

SaveFile приймає назву файла та його вміст, формує службовий ідентифікатор, виконує демонстраційну обробку вмісту, створює LocalFileRecord і повертає fileId. TryReadFile працює з ідентифікатором файла, намагається знайти запис, відновити вміст і повернути результат читання через true/false та параметр content.

Криптографічна обробка файлового вмісту подається через інтерфейс IStorageCryptoService і демонстраційну реалізацію DemoStorageCryptoService. Методи Protect і TryUnprotect використовуються як частина демонстраційного сценарію. ContentHash розглядається як службове контрольне значення, а не як криптографічна гарантія цілісності.

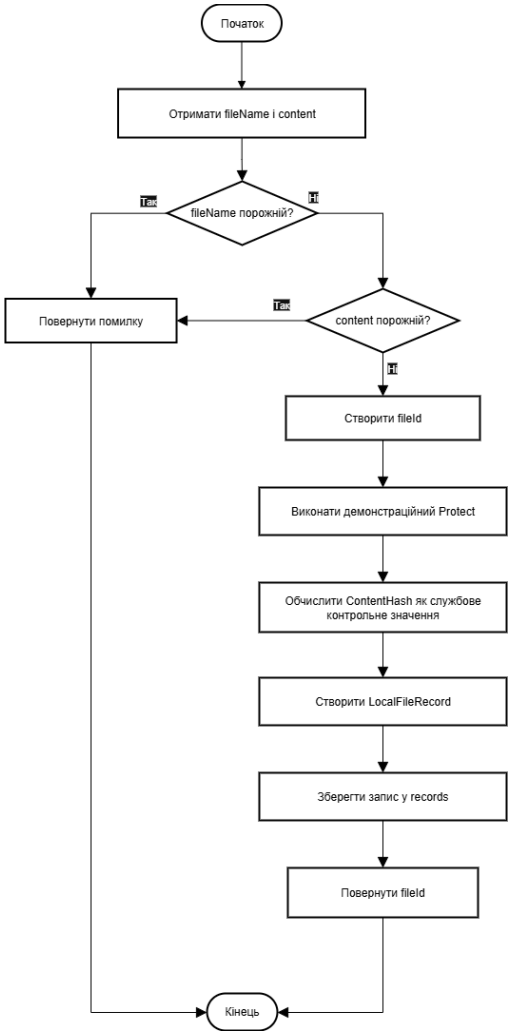

Рис. 3.8 UML activity-діаграма збереження файлових даних

Activity-діаграма для SaveFile показує логіку переходу від вхідних даних до службового запису файла. Вона не описує фізичний шлях, каталог, БД або конкретний файловий формат.

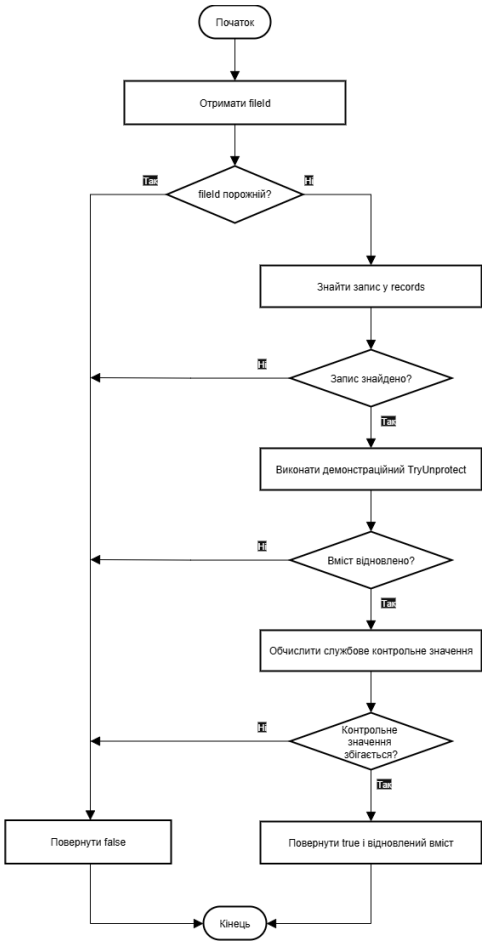

Рис. 3.9 UML activity-діаграма читання файлових даних

Activity-діаграма для TryReadFile показує спробу читання файлового вмісту за fileId, включно з можливим неуспішним результатом.

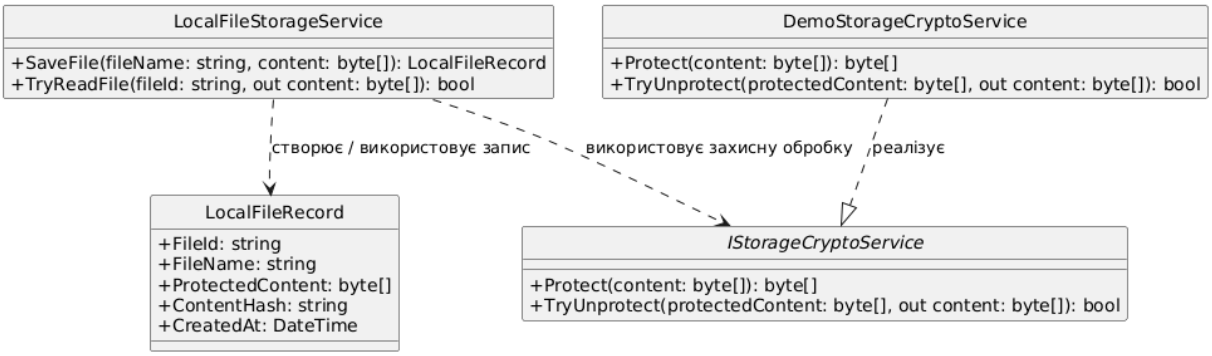

Рис. 3.10 UML class-діаграма фрагмента локального збереження файлових даних

Class-діаграма показує зв’язок між LocalFileStorageService, LocalFileRecord та IStorageCryptoService. Вона не додає фізичну структуру сховища або конкретну технологію БД.

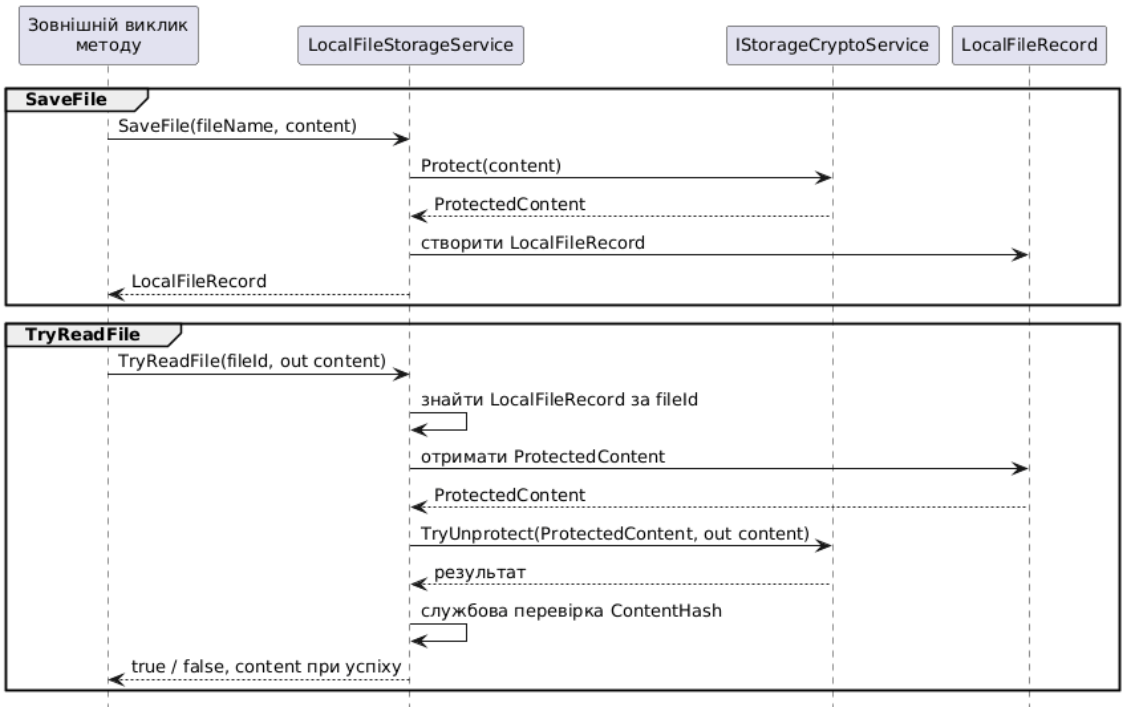

Рис. 3.11 UML sequence-діаграма фрагмента локального збереження файлових даних

Sequence-діаграма пояснює порядок викликів під час збереження та читання файлових даних. Вона не повинна містити файлову систему, БД або конкретний шлях збереження.

Лістинг 3.3 Фрагменти методів SaveFile і TryReadFile

```
public string SaveFile(string fileName, byte[] content)
{
    if (string.IsNullOrWhiteSpace(fileName))
        throw new InvalidOperationException("Назва файла відсутня.");
    if (content == null || content.Length == 0)
        throw new InvalidOperationException("Вміст файла відсутній.");
    var fileId = Guid.NewGuid().ToString("N");
    var protectedContent = _cryptoService.Protect(content);
    var record = new LocalFileRecord
    {
        FileId = fileId,
        FileName = fileName,
        ProtectedContent = protectedContent,
        ContentHash = CalculateContentHash(content),
        CreatedAt = DateTime.UtcNow
    };
    _records[fileId] = record;
```

```
    return fileId;
}
public bool TryReadFile(string fileId, out byte[] content)
{
    content = Array.Empty<byte>();
    if (string.IsNullOrWhiteSpace(fileId))
        return false;
    if (!_records.TryGetValue(fileId, out var record))
        return false;
    if (!_cryptoService.TryUnprotect(record.ProtectedContent, out var restored))
        return false;
    if (record.ContentHash != CalculateContentHash(restored))
        return false;
    content = restored;
    return true;
}
private static string CalculateContentHash(byte[] content)
{
    unchecked
    {
        var value = 17;
        foreach (var item in content)
            value = value * 31 + item;
        return value.ToString("X8");
    }
}
```

У лістингу подано компактні фрагменти SaveFile, TryReadFile і CalculateContentHash. Код показує формування fileId, ProtectedContent, ContentHash, створення LocalFileRecord, читання запису та перевірку службового контрольного значення. Повний код фрагмента винесено в додаток В.

Таблиця 3.5

Демонстраційна перевірка фрагмента локального збереження файлових даних

№	**Сценарій перевірки**	**Вхідні дані / умова**	**Очікуваний результат**	**Статус підтвердження**
1	Збереження файла з коректними даними	fileName заповнений, content не порожній	Формується fileId, створюється LocalFileRecord із ProtectedContent, ContentHash і CreatedAt, метод повертає fileId	Відповідає логіці фрагмента коду
2	Збереження без назви файла	fileName порожній або складається з пробілів	Генерується InvalidOperationException із повідомленням про відсутню назву файла	Відповідає логіці фрагмента коду
3	Збереження без	content == null	Генерується	Відповідає логіці

№	Сценарій перевірки	Вхідні дані / умова	Очікуваний результат	Статус підтвердження
	вмісту файла	або content.Length == 0	InvalidOperationException із повідомленням про відсутній вміст файла	фрагмента коду
4	Читання з порожнім ідентифікатором	fileId порожній або складається з пробілів	Метод повертає false, content залишається порожнім масивом	Відповідає логіці фрагмента коду
5	Читання невідомого файла	У реєстрі відсутній запис із переданим fileId	Метод повертає false	Відповідає логіці фрагмента коду
6	Невдала спроба відновлення вмісту	TryUnprotect повертає false	Метод повертає false без повернення файлового вмісту	Відповідає логіці фрагмента коду
7	Невідповідність службового контрольного значення	ContentHash запису не збігається з CalculateContentHash(restored)	Метод повертає false; ContentHash не подається як криптографічна гарантія	Відповідає логіці фрагмента коду
8	Успішне читання файла	Запис знайдено, вміст відновлено, службове контрольне значення збігається	Метод повертає true, у content передається відновлений вміст	Відповідає логіці фрагмента коду

Таблиця 3.5 показує демонстраційні сценарії перевірки локального збереження файлових даних. Сценарій із ContentHash формулюється як перевірка службового контрольного значення, без твердження про криптографічну гарантію.

3.5 Узагальнення тестування та аналіз результатів роботи програмних фрагментів

Після опису окремих програмних фрагментів узагальнюється їх демонстраційна перевірка. У цьому підрозділі не створюються нові діаграми або кодові фрагменти. Основна увага приділяється зведенню результатів уже описаних сценаріїв.

Перевірка не подається як повне тестування системи. У межах роботи не наводяться фактичні логи запуску, числові метрики продуктивності, відсотки

успішності, покриття коду, CI/CD або використання тестових фреймворків. Таблиця 3.5 виконує роль узагальнювального матеріалу: вона збирає перевірки фрагментів 3.2, 3.3 і 3.4 в одну структуру.

Таблиця 3.6

Зведена таблиця перевірки програмних фрагментів

№	**Фрагмент**	**Сценарій / перевірювана дія**	**Очікуваний результат**	**Статус підтвердження**
1	Прив’язка пристрою	Коректний запит прив’язки	Створено запис прив’язки і повернуто успішний результат	Відповідає логіці фрагмента
2	Прив’язка пристрою	Відсутній або некоректний запит	Повернуто неуспішний результат без створення запису	Відповідає логіці фрагмента
3	Прив’язка пристрою	Відсутній ідентифікатор користувача або пристрою	Повернуто неуспішний результат із поясненням причини	Відповідає логіці фрагмента
4	Прив’язка пристрою	Відсутній PublicKey або AttestationToken	Повернуто неуспішний результат без прив’язки пристрою	Відповідає логіці фрагмента
5	Прив’язка пристрою	Повторна прив’язка вже наявного пристрою	Повернуто неуспішний результат про вже прив’язаний пристрій	Відповідає логіці фрагмента
6	Обмін повідомленнями	Створення пакета з коректного запиту	Сформовано EncryptedMessage Packet і збережено стан Sent	Відповідає логіці фрагмента
7	Обмін повідомленнями	Спроба створення пакета без тексту	Обробка припиняється з помилкою про відсутній текст	Відповідає логіці фрагмента
8	Обмін повідомленнями	Отримання відсутнього пакета	Повернуто неуспішний MessageProcessin gResult	Відповідає логіці фрагмента
9	Обмін повідомленнями	Повторне отримання пакета з уже обробленим	Повернуто неуспішний результат без	Відповідає логіці фрагмента

№	Фрагмент	Сценарій / перевірювана дія	Очікуваний результат	Статус підтвердження
		MessageId	повторного збереження	
10	Обмін повідомленнями	Пакет із некоректною службовою перевіркою	Повернуто неуспішний результат перевірки службових даних	Відповідає логіці фрагмента
11	Обмін повідомленнями	Неможливість відновити CipherText	Повернуто неуспішний результат без збереження PlainText	Відповідає логіці фрагмента
12	Обмін повідомленнями	Коректне отримання пакета	Збережено PlainText і сформовано MessageReceipt зі станом Delivered	Відповідає логіці фрагмента
13	Локальне збереження файлових даних	Збереження файла з коректними даними	Створено LocalFileRecord і повернуто fileId	Відповідає логіці фрагмента
14	Локальне збереження файлових даних	Спроба збереження без назви або вмісту	Обробка припиняється з помилкою про відсутні дані	Відповідає логіці фрагмента
15	Локальне збереження файлових даних	Спроба читання з порожнім або невідомим fileId	Метод повертає false без повернення вмісту	Відповідає логіці фрагмента
16	Локальне збереження файлових даних	Невдала спроба відновлення або невідповідність ContentHash	Метод повертає false; ContentHash трактується лише як службове контрольне значення	Відповідає логіці фрагмента
17	Локальне збереження файлових даних	Успішне читання локально збережених файлових даних	Метод повертає true і передає відновлений content	Відповідає логіці фрагмента

Таблиця 3.6 об’єднує результати демонстраційної перевірки фрагментів прив’язки пристрою, захищеного обміну повідомленнями та локального збереження файлових даних. Окремі рядки цієї таблиці потребують ручної звірки з таблицями відповідних підрозділів перед фінальним оформленням.

Зведення результатів дає змогу узагальнити логіку роботи описаних фрагментів на рівні демонстраційної реалізації. Такий результат не є

доведенням промислової готовності системи, але дозволяє подати обрані фрагменти як взаємопов’язані частини практичної реалізації.

3.6 Вимоги до апаратного та програмного забезпечення

Для виконання описаних програмних фрагментів потрібно визначити умови апаратного та програмного забезпечення. Цей підрозділ не описує серверну інфраструктуру, базу даних, endpoint-и, порти, протоколи або реалізацію мережевого транспорту.

Вимоги подаються в одній об’єднаній таблиці без поділу на користувача і розробника. Такий формат відповідає межам фрагментарної реалізації та не потребує додаткового опису сценаріїв встановлення, розгортання або підтримки.

Таблиця 3.7

Мінімальні та рекомендовані вимоги до апаратного і програмного забезпечення

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**	**Пояснення / обмеження**
Операційна система	Windows 11	Windows 11	-
Оперативна пам’ять	6 ГБ RAM	8 ГБ RAM	6 ГБ подано як робоче мінімальне значення за формулою користувача
Процесор	Окрема вимога до потужного процесора не визначається	Не визначено	Модель, частота і кількість ядер не вказуються
Середовище розробки	Visual Studio 2022	Visual Studio 2022	Редакція Visual Studio не уточнюється
Платформа виконання / розробки	.NET 8	.NET 8	Точний build SDK/runtime не вказується
Тип застосунку	WinForms	WinForms	Тип застосунку узгоджено з реалізаційними фрагментами
Мова програмування	C#	C#	Мова відповідає поданим фрагментам коду
Мережеве підключення	Потрібне для обміну повідомленнями та	Потрібне для обміну повідомленнями та	Endpoint-и, порти, протоколи та серверні

Параметр	Мінімальні вимоги	Рекомендовані вимоги	Пояснення / обмеження
	реєстрації/входу	реєстрації/входу	адреси не визначаються

У таблиці збережені підтверджені позиції: Windows 11 без редакції, мінімальне значення RAM 6 ГБ як робоче значення за формулою користувача, відсутність вимоги до потужного процесора, Visual Studio 2022, .NET 8, WinForms, C#, а також мережеве підключення для обміну повідомленнями та реєстрації/входу.

Вимоги не є результатом продуктивнісного тестування і не підтверджують роботу системи на будь-якому обладнанні. Вони окреслюють умови середовища, у межах якого розглядаються програмні фрагменти.

3.7 Склад інсталяційного пакета

Після визначення вимог до апаратного та програмного забезпечення потрібно окремо зафіксувати склад матеріалів, які могли б входити до інсталяційного пакета повної версії програмної системи. У межах цієї роботи фактичний інсталяційний пакет не створювався, оскільки практична частина розкриває не завершений застосунок, а окремі програмні фрагменти.

У підрозділі розглядається не готовий інсталятор, а орієнтовний склад ZIP-пакета повної версії програмної системи. Такий підхід показує, які групи матеріалів доцільно передбачити у складі пакета, але не створює твердження про фактично сформований архів, готовий виконуваний файл або завершений комплект для встановлення. Процес встановлення або порядок використання ZIP-пакета не описується.

Таблиця 3.8

Орієнтовний склад ZIP-пакета повної версії програмної системи

Елемент пакета	Призначення	Статус включення	Обмеження
Виконуваний файл повної версії застосунку	Основний файл для роботи повної версії програмної системи	Передбачається для повної версії	Конкретна назва файла не вказується; фактичне створення .exe у межах роботи не

Елемент пакета	Призначення	Статус включення	Обмеження
			підтверджується
Супровідні файли виконання	Допоміжні матеріали, потрібні для роботи виконуваної частини повної версії	Передбачається без деталізації складу	Не деталізуються DLL, runtime, тип публікації, .deps.json, .runtimeconfig.json
README / супровідна інструкція	Коротке пояснення складу пакета та загальних умов його підготовки	Передбачається як допоміжний елемент	Конкретна назва файла не вказується; процес встановлення або використання ZIP не описується
Повний код програмних фрагментів	Надання коду реалізованих у роботі програмних фрагментів	Передбачається	Не подається як повний Visual Studio-проєкт
Файли Visual Studio-проєкту	Матеріали середовища розробки, які не входять до складу орієнтовного ZIP-пакета	Не включаються	Не подаються структура проєкту, файли проєкту або службові папки середовища розробки
Демонстраційні або тестові файли	Матеріали перевірки, які не переносяться до складу ZIP-пакета	Не включаються	Не додаються sample-data, test-data, demo-data, журнали запуску або тестові сценарії як файли пакета
Конфігураційні файли	Окрема група налаштувань, яка не розглядається як складова пакета	Не включаються	Не додаються конфігурації, адреси серверів, ключі або локальні шляхи
Серверна частина та база даних	Компоненти, що не входять до меж орієнтовного ZIP-пакета цього підрозділу	Не включаються	Не додаються backend, API, endpoint-и, порти, протоколи, СУБД або файли бази даних
Криптографічні ключі та сертифікати	Матеріали криптографічної інфраструктури, які не входять до складу пакета	Не включаються	Не додаються ключі, сертифікати, сховища ключів або матеріали промислового криптографічного розгортання

Таблиця фіксує орієнтовний склад ZIP-пакета для повної версії програмної системи. Елементи зі статусом “передбачається” не подаються як фактично створені матеріали, а елементи зі статусом “не включаються” задають межі підрозділу.

Окремо розмежовується повний код програмних фрагментів і файли Visual Studio-проєкту. У складі орієнтовного ZIP-пакета може бути передбачено

код фрагментів, описаних у розділі 3, але це не означає включення повного проєкту, службових файлів середовища розробки або структури готової збірки.

Отже, підрозділ 3.8 не підтверджує створення інсталяційного пакета. Він визначає орієнтовний склад ZIP-пакета повної версії програмної системи й фіксує межі, які не дозволяють перетворити опис фрагментарної реалізації на опис повного промислового розгортання.

3.8 Висновки до розділу 3

У третьому розділі розглянуто практичну частину роботи на рівні окремих програмних фрагментів. Основну увагу приділено опису реалізаційних рішень, а не поданню повністю завершеного застосунку.

У межах розділу визначено організацію програмної реалізації та інструментарій розробки. Окремо розглянуто фрагменти, пов’язані з прив’язкою пристрою, захищеним обміном повідомленнями та локальним збереженням файлових даних. Для прив’язки пристрою показано логіку обробки відкритого ключа та атестаційного токена. Для обміну повідомленнями розкрито формування й отримання захищеного пакета. Локальне збереження файлових даних подано без розкриття фізичного формату сховища.

Криптографічна обробка в реалізаційних фрагментах має демонстраційний характер. Вона використовується для пояснення логіки захищеної обробки даних, але не подається як промислова криптографічна інфраструктура і не супроводжується твердженнями про доведену криптографічну стійкість.

Також у розділі узагальнено демонстраційну перевірку програмних фрагментів, сформульовано умови апаратного та програмного забезпечення і визначено орієнтовний склад ZIP-пакета для повної версії програмної системи. Сам пакет у межах роботи не створювався.

Отже, результатом розділу 3 є практичний опис окремих реалізаційних фрагментів, їх демонстраційної перевірки, умов апаратного та програмного

забезпечення і орієнтовного складу пакета для повної версії. Розглянуті фрагменти не суперечать загальному теоретичному обґрунтуванню роботи та концептуальній моделі, поданій у розділі 2.

Водночас розділ 3 не підтверджує створення повної промислової системи. Його зміст обмежується фрагментарною реалізацією без додавання непідтвердженої серверної інфраструктури, бази даних, фізичного формату локального сховища або готового інсталяційного пакета.

ВИСНОВОКИ

У дипломній роботі розглянуто концептуальну модель та фрагменти програмного забезпечення захищеного десктопного месенджера. Основну увагу приділено персональному обміну повідомленнями та файлами у форматі один-на-один, локальному зберіганню основного контенту, службовій ролі серверного контуру та контролю активного пристрою користувача.

У першому розділі виконано системний аналіз предметної області захищених десктопних месенджерів. Визначено основні вимоги до системи, ризики та обмеження запропонованої моделі. Окремо розглянуто наявні рішення у сфері захищених месенджерів, зокрема Signal, Telegram, WhatsApp, Element/Matrix і Threema. На основі аналізу сформульовано постановку завдання для подальшого концептуального моделювання.

У другому розділі побудовано концептуальну модель захищеного десктопного месенджера. Визначено межі системи, ролі учасників, логічну модель даних, організацію інформаційної бази, правила доступу, процес захищеного обміну повідомленнями та файлами, життєві цикли повідомлення і запиту глобального видалення, а також модель захищеності. У моделі збережено розмежування локального довіреного контуру, серверного службового контуру та корпоративного контуру.

У третьому розділі розглянуто практичну частину роботи на рівні окремих програмних фрагментів. Описано фрагмент прив’язки пристрою, фрагмент створення та обробки захищеного пакета повідомлення, а також фрагмент локального збереження файлових даних. Для цих фрагментів наведено код, UML-діаграми та демонстраційні сценарії перевірки.

Практична реалізація має фрагментарний характер. У роботі не підтверджується створення повної промислової системи, завершеної серверної інфраструктури, фізичної бази даних, повного криптографічного протоколу або готового інсталяційного пакета. Криптографічна обробка в програмних фрагментах використовується для демонстрації логіки захищеної обробки даних і не подається як промислова криптографічна інфраструктура.

Поставлену мету дипломної роботи досягнуто: сформовано концептуальну модель захищеного десктопного месенджера та розглянуто програмні фрагменти, які демонструють окремі механізми її реалізації. Виконані завдання охоплюють аналіз предметної області, визначення вимог, аналіз ризиків і обмежень, огляд існуючих рішень, побудову концептуальної моделі та опис фрагментів програмного забезпечення.

Отримані результати можуть бути використані як основа для подальшого розвитку системи: деталізації серверного контуру, фізичної структури бази даних, повного криптографічного протоколу, механізмів резервного копіювання, розгортання та підготовки повного інсталяційного пакета.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Signal Messenger. Linked Devices // Signal Support https://support.signal.org/hc/en-us/articles/360007320551-Linked-Devices
2. Signal Messenger. Backup and Restore Messages // Signal Support https://support.signal.org/hc/en-us/articles/360007059752-Backup-and-Restore-Messages
3. Telegram. Telegram FAQ: Secret Chats // Telegram https://telegram.org/faq
4. Telegram. Introducing Telegram Business // Telegram Blog. 2024 https://telegram.org/blog/telegram-business
5. WhatsApp. About end-to-end encryption // WhatsApp Help Center https://faq.whatsapp.com/general/security-and-privacy/end-to-end-encryption
6. WhatsApp. About linked devices // WhatsApp Help Center https://faq.whatsapp.com/378279804439436
7. WhatsApp. WhatsApp Business Platform: Business messaging APIs // WhatsApp Business https://business.whatsapp.com/products/business-platform
8. Element. Secure collaboration and messaging // Element https://element.io/
9. Matrix.org Foundation. Matrix Specification // Matrix Specification https://spec.matrix.org/
10. Threema. Which data gets stored at Threema? // Threema FAQ https://threema.com/en/faq/data
11. Threema. Threema OnPrem // Threema https://threema.com/en/products/onprem
12. Dworkin M. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST Special Publication 800-38D.

National Institute of Standards and Technology, 2007. DOI: 10.6028/NIST.SP.800-38D.

13. National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. FIPS PUB 180-4. NIST, 2015. DOI: 10.6028/NIST.FIPS.180-4.
14. Moriarty K., Kaliski B., Jonsson J., Rusch A. *PKCS #1: RSA Cryptography Specifications Version 2.2*. RFC 8017. RFC Editor, 2016. DOI: 10.17487/RFC8017.
15. Barker E. *Recommendation for Key Management: Part 1 – General*. NIST Special Publication 800-57 Part 1 Revision 5. National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.SP.800-57pt1r5.
16. OWASP Foundation. *Cryptographic Storage Cheat Sheet*. OWASP Cheat Sheet Series. Дата звернення: 24.05.2026.
17. Temoshok D. та ін. *Digital Identity Guidelines: Authentication and Authenticator Management*. NIST Special Publication 800-63B-4. National Institute of Standards and Technology, 2025. DOI: 10.6028/NIST.SP.800-63B-4.
18. Joint Task Force. *Security and Privacy Controls for Information Systems and Organizations*. NIST Special Publication 800-53 Revision 5. National Institute of Standards and Technology, 2020. DOI: 10.6028/NIST.SP.800-53r5.

Додаток А

Повний код фрагмента прив’язки пристрою

```
using System;
using System.Collections.Generic;
// Код до підрозділу 3.2: фрагмент прив’язки пристрою.
// Демонстраційна реалізація для розділу 3 дипломної роботи.
namespace DiplomaFragments.DeviceBinding
{
    public sealed class DeviceBindingService
    {
        private readonly IDeviceBindingRepository _repository;
        public DeviceBindingService(IDeviceBindingRepository repository)
        {
            _repository = repository;
        }
        public DeviceBindingResult BindDevice(DeviceBindingRequest request)
        {
            if (request == null)
                return DeviceBindingResult.Fail("Запит прив'язки відсутній.");
            if (string.IsNullOrWhiteSpace(request.UserId))
                return DeviceBindingResult.Fail("Не вказано користувача.");
            if (string.IsNullOrWhiteSpace(request.DeviceId))
                return DeviceBindingResult.Fail("Не вказано пристрій.");
            if (string.IsNullOrWhiteSpace(request.PublicKey))
                return DeviceBindingResult.Fail("Відкритий ключ відсутній.");
            if (string.IsNullOrWhiteSpace(request.AttestationToken))
                return DeviceBindingResult.Fail("Атестаційний токен відсутній.");
            if (_repository.Exists(request.UserId, request.DeviceId))
                return DeviceBindingResult.Fail("Пристрій уже прив'язано.");
            var binding = DeviceBinding.Create(
                request.UserId,
                request.DeviceId,
                request.PublicKey,
                request.AttestationToken);
            _repository.Save(binding);
            return DeviceBindingResult.Success(binding.BindingId);
        }
    }
    public sealed record DeviceBindingRequest(
        string UserId,
        string DeviceId,
        string PublicKey,
        string AttestationToken);
    public sealed class DeviceBinding
    {
        public string BindingId { get; private init; } = string.Empty;
        public string UserId { get; private init; } = string.Empty;
        public string DeviceId { get; private init; } = string.Empty;
        public string PublicKey { get; private init; } = string.Empty;
        public string AttestationToken { get; private init; } = string.Empty;
        public DateTime CreatedAt { get; private init; }
        public static DeviceBinding Create(
            string userId,
            string deviceId,
            string publicKey,
            string attestationToken)
        {
            return new DeviceBinding
            {
                BindingId = Guid.NewGuid().ToString("N"),
```

```
                UserId = userId,
                DeviceId = deviceId,
                PublicKey = publicKey,
                AttestationToken = attestationToken,
                CreatedAt = DateTime.UtcNow
            };
        }
    }
    public sealed class DeviceBindingResult
    {
        private DeviceBindingResult(bool success, string? bindingId, string?
error)
        {
            Success = success;
            BindingId = bindingId;
            Error = error;
        }
        public bool Success { get; }
        public string? BindingId { get; }
        public string? Error { get; }
        public static DeviceBindingResult Success(string bindingId)
        {
            return new DeviceBindingResult(true, bindingId, null);
        }
        public static DeviceBindingResult Fail(string error)
        {
            return new DeviceBindingResult(false, null, error);
        }
    }
    public interface IDeviceBindingRepository
    {
        bool Exists(string userId, string deviceId);
        void Save(DeviceBinding binding);
    }
    public sealed class InMemoryDeviceBindingRepository :
IDeviceBindingRepository
    {
        private readonly Dictionary<string, DeviceBinding> _items = new();
        public bool Exists(string userId, string deviceId)
        {
            var key = BuildKey(userId, deviceId);
            return _items.ContainsKey(key);
        }
        public void Save(DeviceBinding binding)
        {
            var key = BuildKey(binding.UserId, binding.DeviceId);
            _items[key] = binding;
        }
        private static string BuildKey(string userId, string deviceId)
        {
            return $"{userId}:{deviceId}";
        }
    }
}
```

Додаток Б

Повний код фрагмента захищеного обміну повідомленнями

```
using System;
using System.Collections.Generic;
using System.Text;
// Код до підрозділу 3.3: фрагмент захищеного обміну повідомленнями.
// Криптографічна обробка має демонстраційний характер і не є промисловою
криптографічною інфраструктурою.
namespace DiplomaFragments.MessageExchange
{
    public sealed class MessageExchangeService
    {
        private readonly IMessageCryptoService _crypto;
        private readonly ILocalMessageStore _store;
        private readonly ReceiptService _receiptService;
        public MessageExchangeService(
            IMessageCryptoService crypto,
            ILocalMessageStore store,
            ReceiptService receiptService)
        {
            _crypto = crypto;
            _store = store;
            _receiptService = receiptService;
        }
        public EncryptedMessagePacket CreatePacket(MessageCreateRequest request)
        {
            if (request == null)
                throw new ArgumentNullException(nameof(request));
            if (string.IsNullOrWhiteSpace(request.PlainText))
                throw new InvalidOperationException("Текст повідомлення
відсутній.");
            var packet = new EncryptedMessagePacket
            {
                MessageId = Guid.NewGuid().ToString("N"),
                SenderDeviceId = request.SenderDeviceId,
                ReceiverDeviceId = request.ReceiverDeviceId,
                CipherText = _crypto.Protect(request.PlainText),
                CreatedAt = DateTime.UtcNow
            };
            packet.Signature = _crypto.Sign(BuildSignatureData(packet));
            _store.SaveState(packet.MessageId, MessageState.Sent);
            return packet;
        }
        public MessageProcessingResult ReceivePacket(EncryptedMessagePacket
packet)
        {
            if (packet == null)
                return Failure("Пакет повідомлення відсутній.");
            if (_store.Exists(packet.MessageId))
                return Failure("Повідомлення з таким ідентифікатором уже
оброблено.");
            if (!_crypto.Verify(BuildSignatureData(packet), packet.Signature))
                return Failure("Службові дані пакета не пройшли перевірку.");
            if (!_crypto.TryUnprotect(packet.CipherText, out var plainText))
                return Failure("Вміст пакета не вдалося відновити.");
            _store.SaveReceived(packet.MessageId, plainText);
            var receipt =
_receiptService.CreateDeliveredReceipt(packet.MessageId);
            return MessageProcessingResult.Success(plainText, receipt);
        }
```

```
        private static string BuildSignatureData(EncryptedMessagePacket packet)
        {
            return string.Join("|", packet.MessageId, packet.SenderDeviceId,
                packet.ReceiverDeviceId, packet.CipherText,
packet.CreatedAt.Ticks);
        }
        private static MessageProcessingResult Failure(string reason)
        {
            return MessageProcessingResult.Failure(reason);
        }
    }
    public sealed record MessageCreateRequest(
        string SenderDeviceId,
        string ReceiverDeviceId,
        string PlainText);
    public sealed class EncryptedMessagePacket
    {
        public string MessageId { get; set; } = string.Empty;
        public string SenderDeviceId { get; set; } = string.Empty;
        public string ReceiverDeviceId { get; set; } = string.Empty;
        public string CipherText { get; set; } = string.Empty;
        public string Signature { get; set; } = string.Empty;
        public DateTime CreatedAt { get; set; }
    }
    public sealed class MessageProcessingResult
    {
        private MessageProcessingResult(
            bool success,
            string? plainText,
            MessageReceipt? receipt,
            string? error)
        {
            Success = success;
            PlainText = plainText;
            Receipt = receipt;
            Error = error;
        }
        public bool Success { get; }
        public string? PlainText { get; }
        public MessageReceipt? Receipt { get; }
        public string? Error { get; }
        public static MessageProcessingResult Success(string plainText,
MessageReceipt receipt)
        {
            return new MessageProcessingResult(true, plainText, receipt, null);
        }
        public static MessageProcessingResult Failure(string error)
        {
            return new MessageProcessingResult(false, null, null, error);
        }
    }
    public sealed record MessageReceipt(
        string MessageId,
        MessageState State,
        DateTime CreatedAt);
    public enum MessageState
    {
        Sent,
        Delivered
    }
    public sealed class ReceiptService
    {
        public MessageReceipt CreateDeliveredReceipt(string messageId)
        {
```

```
            return new MessageReceipt(messageId, MessageState.Delivered,
DateTime.UtcNow);
        }
    }
    public interface IMessageCryptoService
    {
        string Protect(string plainText);
        bool TryUnprotect(string protectedText, out string plainText);
        string Sign(string data);
        bool Verify(string data, string signature);
    }
    public sealed class SimpleMessageCryptoService : IMessageCryptoService
    {
        public string Protect(string plainText)
        {
            var bytes = Encoding.UTF8.GetBytes(plainText);
            return Convert.ToBase64String(bytes);
        }
        public bool TryUnprotect(string protectedText, out string plainText)
        {
            try
            {
                var bytes = Convert.FromBase64String(protectedText);
                plainText = Encoding.UTF8.GetString(bytes);
                return true;
            }
            catch (FormatException)
            {
                plainText = string.Empty;
                return false;
            }
        }
        public string Sign(string data)
        {
            unchecked
            {
                var value = 23;
                foreach (var item in data)
                    value = value * 31 + item;
                return value.ToString("X8");
            }
        }
        public bool Verify(string data, string signature)
        {
            return Sign(data) == signature;
        }
    }
    public interface ILocalMessageStore
    {
        bool Exists(string messageId);
        void SaveState(string messageId, MessageState state);
        void SaveReceived(string messageId, string plainText);
    }
    public sealed class InMemoryMessageStore : ILocalMessageStore
    {
        private readonly Dictionary<string, MessageState> _states = new();
        private readonly Dictionary<string, string> _messages = new();
        public bool Exists(string messageId)
        {
            return _states.ContainsKey(messageId) ||
_messages.ContainsKey(messageId);
        }
        public void SaveState(string messageId, MessageState state)
        {
```

```
            _states[messageId] = state;
        }
        public void SaveReceived(string messageId, string plainText)
        {
            _states[messageId] = MessageState.Delivered;
            _messages[messageId] = plainText;
        }
    }
    public sealed class MessageExchangeScenarioCheck
    {
        public MessageProcessingResult Run()
        {
            var crypto = new SimpleMessageCryptoService();
            var store = new InMemoryMessageStore();
            var receiptService = new ReceiptService();
            var service = new MessageExchangeService(crypto, store,
receiptService);
            var packet = service.CreatePacket(new MessageCreateRequest(
                SenderDeviceId: "sender-device",
                ReceiverDeviceId: "receiver-device",
                PlainText: "Демонстраційне повідомлення"));
            return service.ReceivePacket(packet);
        }
    }
}
```

Додаток В

Повний код фрагмента локального збереження файлових даних

```
using System;
using System.Collections.Generic;
using System.Text;
// Код до підрозділу 3.4: фрагмент локального збереження файлових даних.
// Криптографічна обробка має демонстраційний характер і не є промисловою
криптографічною інфраструктурою.
namespace DiplomaFragments.LocalFileStorage
{
    public sealed class LocalFileStorageService
    {
        private readonly IStorageCryptoService _cryptoService;
        private readonly Dictionary<string, LocalFileRecord> _records = new();
        public LocalFileStorageService(IStorageCryptoService cryptoService)
        {
            _cryptoService = cryptoService;
        }
        public string SaveFile(string fileName, byte[] content)
        {
            if (string.IsNullOrWhiteSpace(fileName))
                throw new InvalidOperationException("Назва файла відсутня.");
            if (content == null || content.Length == 0)
                throw new InvalidOperationException("Вміст файла відсутній.");
            var fileId = Guid.NewGuid().ToString("N");
            var protectedContent = _cryptoService.Protect(content);
            var record = new LocalFileRecord
            {
                FileId = fileId,
                FileName = fileName,
                ProtectedContent = protectedContent,
                ContentHash = CalculateContentHash(content),
                CreatedAt = DateTime.UtcNow
            };
            _records[fileId] = record;
            return fileId;
        }
        public bool TryReadFile(string fileId, out byte[] content)
        {
            content = Array.Empty<byte>();
            if (string.IsNullOrWhiteSpace(fileId))
                return false;
            if (!_records.TryGetValue(fileId, out var record))
                return false;
            if (!_cryptoService.TryUnprotect(record.ProtectedContent, out var
restored))
                return false;
            if (record.ContentHash != CalculateContentHash(restored))
                return false;
            content = restored;
            return true;
        }
        private static string CalculateContentHash(byte[] content)
        {
            unchecked
            {
                var value = 17;
                foreach (var item in content)
                    value = value * 31 + item;
                return value.ToString("X8");
```

```
            }
        }
    }
    public sealed class LocalFileRecord
    {
        public string FileId { get; set; } = string.Empty;
        public string FileName { get; set; } = string.Empty;
        public byte[] ProtectedContent { get; set; } = Array.Empty<byte>();
        public string ContentHash { get; set; } = string.Empty;
        public DateTime CreatedAt { get; set; }
    }
    public interface IStorageCryptoService
    {
        byte[] Protect(byte[] content);
        bool TryUnprotect(byte[] protectedContent, out byte[] content);
    }
    public sealed class DemoStorageCryptoService : IStorageCryptoService
    {
        public byte[] Protect(byte[] content)
        {
            var text = Convert.ToBase64String(content);
            return Encoding.UTF8.GetBytes(text);
        }
        public bool TryUnprotect(byte[] protectedContent, out byte[] content)
        {
            try
            {
                var text = Encoding.UTF8.GetString(protectedContent);
                content = Convert.FromBase64String(text);
                return true;
            }
            catch (FormatException)
            {
                content = Array.Empty<byte>();
                return false;
            }
        }
    }
}
```

Додаток Д

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Масло Серігй Вячеславович здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Концептуальна модель та фрагменти програмного забезпечення захищеного десктопного месенджеру» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що:
    - ○ інструменти ШІ (ChatGPT 5.5) були використані для:
        - ■ [ ] перекладу іншомовних джерел;
        - ■ [x] стилістичного редагування та перевірки граматики;
        - ■ [x] допомоги у написанні/відлагодженні програмного коду;
        - ■ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«17» травня 2026 р. _____Масло_____ **Сергій МАСЛО**
(підпис)