ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка та впровадження багаторівневого захисту мобільного застосунку на базі Android»

зі спеціальності *125 Кібербезпека та захист інформації*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ХОЛДЄЄВ
(підпис)

Виконав: здобувач вищої освіти групи БСДМ-61
ХОЛДЄЄВ Олександр
(прізвище, ім'я)

Керівник д.т.н., професор ЗИБІН Сергій
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА СИСТЕМ ТА ТЕХНОЛОГІЙ КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка та впровадження багаторівневого захисту мобільного застосунку на базі Android»

зі спеціальності *125 Кібербезпека та захист інформації*
(код, найменування спеціальності)

освітньо-професійної програми *Інформаційна та кібернетична безпека*
(назва програми)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Олександр ХОЛДЄЄВ
(підпис)

Виконав: здобувач вищої освіти групи
БСДМ-61
ХОЛДЄЄВ Олександр
(прізвище, ім'я)

Керівник д.т.н., професор ЗИБІН Сергій
(науковий ступінь, вчене звання, прізвище, ім'я)

Рецензент _____
(науковий ступінь, вчене звання, прізвище, ім'я)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

Кафедра Систем та технологій кібербезпеки
Ступінь вищої освіти Магістр
Спеціальність 125 Кібербезпека та захист інформації
Освітньо-професійна програма Інформаційна та кібернетична безпека

ЗАТВЕРДЖУЮ
Завідувач кафедри
Систем та технологій
кібербезпеки
Галина
_____ ГАЙДУР
“___” жовтня 2025
року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

ХОЛДЄЄВ Олександр Олександрович
(прізвище, ім’я)

1. Тема кваліфікаційної роботи: « Розробка та впровадження багаторівневого захисту мобільного застосунку на базі Android»

керівник кваліфікаційної роботи д.т.н., професор Зибін Сергій Вікторович
(прізвище, ім’я, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» жовтня 2025 року № ___.

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 15.12.2025 р.

3. Вихідні дані до кваліфікаційної роботи вимоги та рекомендації з безпеки мобільних Android застосунків,

хмарна інфраструктура Firebase та програмні інструменти для розробки і тестування захищеного мобільного застосунку.

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

- Аналіз загроз інформаційній безпеці мобільних Android-застосунків.
- Дослідження та вибір методів багаторівневого захисту мобільного застосунку.
- Розробка та реалізація захищеного мобільного застосунку для обміну повідомленнями.
- Тестування й оцінка ефективності реалізованих механізмів безпеки.

5. Перелік графічного матеріалу

Презентація PowerPoint.

6. Дата видачі завдання 01.10.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз наукових і технічних джерел з питань інформаційної безпеки мобільних Android-застосунків.	01.10.2025 р.	
2.	Дослідження сучасних загроз і механізмів багаторівневого захисту мобільних застосунків.	12.10.2025 р.	

3.	Проєктування архітектури безпечного мобільного застосунку та формування моделі загроз.	27.10.2025 р.	
4.	Розробка прототипу мобільного Android-застосунку з реалізацією багаторівневого захисту.	03.11.2025 р.	
5.	Впровадження та налаштування механізмів захисту коду, даних, автентифікації та мережевих з’єднань.	15.11.2025 р.	
6.	Проведення тестування системи безпеки та аналіз отриманих результатів.	26.11.2025 р.	
7.	Оформлення пояснювальної записки та підготовка роботи до захисту.	15.12.2025 р.	

Здобувач вищої освіти _____ Олександр ХОЛДЄЄВ
(підпис) *(ім’я, прізвище)*

Керівник кваліфікаційної роботи _____ Сергій ЗИБІН
(підпис) *(ім’я, прізвище)*

РЕФЕРАТ

Магістерська дисертація присвячена дослідженню, розробці та впровадженню багаторівневого захисту мобільного додатку на базі операційної системи Android. Актуальність теми зумовлена стрімким зростанням кількості мобільних додатків, що обробляють конфіденційні дані користувачів, а також зростанням рівня кіберзагроз, спрямованих на мобільні платформи.

Мета роботи: підвищення рівня інформаційної безпеки мобільного Android-додатку шляхом застосування комплексного багаторівневого підходу до захисту коду, даних, мережевих з'єднань та механізмів автентифікації користувачів.

Об'єкт дослідження: процес забезпечення інформаційної безпеки мобільних додатків.

Предмет дослідження: методи та засоби реалізації багаторівневого захисту мобільного додатку на платформі Android.

Методи дослідження: підвищення рівня інформаційної безпеки мобільного Android-додатку шляхом застосування комплексного багаторівневого підходу до захисту коду, даних, мережевих з'єднань та механізмів автентифікації користувачів.

Короткий зміст роботи: практична частина роботи реалізована у вигляді прототипу мобільного додатку для обміну текстовими повідомленнями в режимі реального часу. У процесі розробки використовувалися мова програмування Kotlin, фреймворк Jetpack Compose та хмарні сервіси Firebase. Для забезпечення конфіденційності даних було реалізовано наскрізне шифрування повідомлень з використанням сучасних криптографічних алгоритмів та механізмів безпечного зберігання ключів Android Keystore. Додатково було реалізовано безпечну автентифікацію користувачів, контроль доступу до даних та захист клієнтського коду за допомогою обфускації.

Розроблений додаток було протестовано та оцінено його відповідність рекомендаціям OWASP Mobile Top 10. Отримані результати підтверджують ефективність запропонованого багаторівневого підходу до захисту та можливість його практичного застосування під час розробки безпечних мобільних систем.

Магістерська дисертація містить 62 сторінки, 5 рисунків, 36 джерел у списку літератури.

Ключові слова: Android, мобільний додаток, інформаційна безпека, багаторівневий захист, шифрування, Firebase, автентифікація, кібербезпека.

ABSTRACT

This master's qualification thesis is devoted to the research, development, and implementation of a multi-layer security system for a mobile application based on the Android operating system. The relevance of the topic is determined by the rapid growth of mobile applications that process confidential user data and by the increasing number of cyber threats targeting mobile platforms.

The aim of this work is to improve the level of information security of an Android mobile application through the implementation of a comprehensive multi-layer protection approach, including secure authentication, data protection, network security, and client-side code protection.

The object of the research is the process of ensuring information security of mobile applications.

The subject of the research is the methods, tools, and technologies used to implement multi-layer security in Android-based mobile applications.

The thesis analyzes the main security threats to mobile applications and examines modern approaches to mobile cybersecurity. Special attention is given to the architectural principles of secure Android application development based on the recommendations of OWASP Mobile Top 10 and official Android security guidelines.

The practical part of the work is implemented as a prototype of a real-time mobile messaging application. The application is developed using Kotlin, Jetpack Compose, and Firebase cloud services. To ensure data confidentiality, end-to-end encryption is implemented using modern cryptographic algorithms and secure key storage mechanisms provided by the Android Keystore system. Secure user authentication, access control, and client-side code protection techniques are also applied.

The developed application was tested for security vulnerabilities and evaluated according to OWASP Mobile Top 10 recommendations. The results confirm the effectiveness of the proposed multi-layer security approach and demonstrate the feasibility of combining a high level of data protection with acceptable performance and user experience.

Keywords: Android, mobile application, information security, multi-layer protection, encryption, Firebase, authentication, cybersecurity.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ОБСТАВИН

API — інтерфейс прикладного програмування, що забезпечує взаємодію між програмними компонентами.

AES — алгоритм симетричного шифрування Advanced Encryption Standard, що використовується для захисту даних.

Android Keystore — системний сервіс операційної системи Android для безпечного зберігання криптографічних ключів.

Auth (Автентифікація) — процес автентифікації користувача в системі.

Cloud Firestore — хмарна документоорієнтована база даних Firebase з підтримкою синхронізації в реальному часі.

ECDH — алгоритм узгодження криптографічних ключів на основі еліптичної кривої Діффі-Хеллмана.

E2EE (Наскрізне шифрування) — наскрізне шифрування, при якому лише кінцеві користувачі мають доступ до відкритого тексту повідомлень.

Firebase — хмарна платформа Google для розробки та підтримки мобільних додатків.

Authentication (Автентифікація Firebase) — сервіс Firebase для керування автентифікацією та авторизацією користувачів.

Pravila Firestore — правила безпеки, що визначають доступ користувачів до даних у Cloud Firestore.

HTTPS — протокол безпечної передачі гіпертексту.

MFA (Багатофакторна автентифікація) — багатофакторна автентифікація користувача.

MVVM — архітектурний шаблон Model–View–ViewModel для створення клієнтських застосунків.

OWASP — Open Worldwide Application Security Project, міжнародна організація, що займається питаннями безпеки програмного забезпечення.

OWASP Mobile Top 10 — список найпоширеніших загроз безпеці мобільних застосунків.

TLS — криптографічний протокол Transport Layer Security для захисту мережевих з'єднань.

UID — унікальний ідентифікатор користувача.

ViewModel — компонент архітектури Android, призначений для зберігання та керування станом інтерфейсу.

ВСТУП

Актуальність дослідження. Стрімкий розвиток цифрових технологій та широке впровадження мобільних пристроїв у всі сфери суспільного життя призвели до значного зростання ролі мобільних додатків як інструментів для обробки, зберігання та передачі інформації. Сучасні Android-додатки використовуються для особистого та корпоративного спілкування, фінансових операцій, електронного документообігу, зберігання персональних даних, доступу до державних та комерційних послуг. У зв'язку з цим мобільні додатки стають однією з основних цілей кібератак, спрямованих на компрометацію конфіденційної інформації користувачів. Аналіз сучасних наукових публікацій, технічної документації та міжнародних стандартів у сфері мобільної безпеки показує, що значна частина існуючих Android-додатків реалізує лише окремі захисні механізми, такі як базова автентифікація або захист мережевих з'єднань. Такий підхід не забезпечує належного рівня захисту в умовах зростаючої складності та кількості атак. Проблема комплексного впровадження багаторівневого захисту, який би одночасно охоплював захист клієнтського коду, даних, мережевих з'єднань та механізмів автентифікації, а також був адаптований до реальних умов експлуатації мобільних додатків, залишається невирішеною. Наукові завдання Для досягнення мети, поставленої в магістерській кваліфікаційній роботі, необхідно вирішити такі основні наукові завдання:

• проаналізувати сучасні загрози інформаційній безпеці мобільних Android-додатків;

• дослідити існуючі підходи та механізми забезпечення багаторівневого захисту мобільних додатків;

• сформувати архітектурні вимоги до безпечного мобільного додатку;

• розробити та впровадити прототип Android-додатку з комплексною системою захисту;

• оцінити ефективність впроваджених заходів безпеки шляхом тестування та аналізу.

Практичне значення отриманих результатів. полягає в можливості використання розробленого прототипу мобільного додатку та запропонованих архітектурних і програмних рішень під час створення безпечних Android-додатків. Матеріали роботи можуть бути використані в практичній діяльності розробників мобільного програмного забезпечення, а також у навчальному процесі при підготовці фахівців у галузі інформаційної та кібербезпеки.

Апробація результатів. Результати, отримані під час виконання магістерської кваліфікаційної роботи, планується використовувати під час навчального процесу та можуть бути представлені у вигляді доповідей на науково-практичних конференціях або семінарах, присвячених інформаційній безпеці та розробці мобільних додатків.

1 АНАЛІЗ СТАНУ ПРОБЛЕМИ ТА ТЕОРЕТИЧНІ ОСНОВИ

1.1. Загрози інформаційній безпеці мобільних застосунків

Мобільні додатки на платформі Android – це складні програмні системи, що функціонують у динамічному та гетерогенному середовищі. Вони поєднують клієнтську частину, операційну систему пристрою, мережеві канали зв'язку та серверну або хмарну інфраструктуру. У процесі своєї роботи такі додатки обробляють значні обсяги конфіденційної інформації, включаючи персональні дані користувачів, облікові дані автентифікації, історію взаємодії, а у випадку з комунікаційними сервісами – особисті повідомлення. Це призводить до підвищеної уваги до питань інформаційної безпеки мобільних рішень.

Особливістю платформи Android є її відкритість та масштабованість екосистеми. З одного боку, це забезпечує гнучкість у розробці та швидкому впровадженні інновацій, а з іншого – створює широкий спектр потенційних векторів атак. Фрагментація версій операційних систем, різноманітність апаратних конфігурацій, наявність кастомних прошивок та можливість встановлення додатків з неофіційних джерел ускладнюють забезпечення єдиного рівня захисту для всіх пристроїв.

Доцільно класифікувати загрози інформаційній безпеці мобільних додатків за рівнями виникнення. На рівні користувача основними ризиками є фішинг, компрометація облікових даних, використання слабких або повторно використаних паролів, соціальна інженерія та несанкціонований доступ до пристрою. У випадку месенджерів додатковою небезпекою є доступ третіх осіб до розблокованого пристрою або використання облікового запису без повторної автентифікації.

На рівні клієнтського додатку поширеними є загрози, пов'язані зі зворотним проектуванням, модифікацією коду, аналізом байт-коду та підміною логіки виконання. Відсутність обфускації або слабка перевірка цілісності коду спрощує аналіз додатків і може призвести до виявлення прихованих API, ключів або внутрішніх алгоритмів. Такі атаки особливо небезпечні для додатків, які реалізують власні механізми шифрування або контролю доступу.

Окрему групу загроз складають вразливості, пов'язані з неправильним використанням механізмів платформи Android. Прикладами є небезпечне використання неявних намірів, помилки міжпроцесного зв'язку, надмірні дозволи або неправильна взаємодія з постачальниками контенту. У таких випадках сторонні додатки можуть перехоплювати дані або перешкоджати виконанню легітимного додатка.

Мережевий рівень є одним із найважливіших для мобільних додатків. Незахищені або неправильно налаштовані мережеві з'єднання створюють умови для атак типу «людина посередині», перехоплення трафіку, підміни повідомлень або повторного відтворення запитів. Для онлайн-чатів, що працюють у режимі реального часу, мережеві загрози мають особливе значення, оскільки значна частина даних постійно передається між клієнтом і сервером.

На рівні сервера або хмарної інфраструктури основними ризиками є неправильне налаштування системи доступу, витік токенів автентифікації, помилки в правилах доступу до бази даних та надмірні привілеї користувачів. У контексті використання Firebase Firestore типовою загрозою є неправильно налаштовані правила безпеки, які можуть дозволити читання або запис даних користувачами, які не мають відповідних прав.

Згідно з рекомендаціями OWASP Mobile Top 10, найпоширенішими класами загроз для мобільних додатків є незахищене сховище даних, недостатня криптографія, незахищені мережеві з'єднання, модифікація коду на стороні клієнта та помилки автентифікації. Аналіз цих загроз показує, що ефективний захист неможливий за допомогою використання єдиного механізму безпеки, а вимагає комплексного підходу.

1.2. Підходи та механізми до забезпечення кібербезпеки мобільних додатків

Система безпеки мобільних додатків на платформі Android базується на багаторівневій моделі захисту, яка поєднує апаратні, системні та програмні механізми. Фундаментальним принципом є ізоляція додатків, згідно з якою кожен додаток працює у власному процесі та має обмежений доступ до ресурсів інших компонентів системи.

Одним із ключових механізмів безпеки є система дозволів Android. Вона регулює доступ додатків до чутливих ресурсів пристрою, таких як камера, мікрофон, контакти, геолокація та сховище файлів. Впровадження динамічної моделі дозволів збільшило контроль користувачів та зменшило ризики прихованого доступу до персональних даних.

Криптографічні механізми широко використовуються для захисту конфіденційної інформації. Android надає стандартні API для симетричного та асиметричного шифрування, а також спеціалізований сервіс Android Keystore, який дозволяє зберігати криптографічні ключі в захищеному середовищі пристрою. Це значно знижує ймовірність компрометації ключів навіть за умови доступу до файлової системи.

Важливим підходом до кібербезпеки є концепція «глибинно ешелонованого захисту» (багаторівневого захисту), яка передбачає використання кількох незалежних механізмів захисту. У разі компрометації одного рівня, інші рівні продовжують забезпечувати безпеку системи. Для мобільних чатів це означає поєднання безпечної автентифікації, шифрування даних, захисту мережі та контролю доступу. Окрему роль відіграють механізми автентифікації та авторизації користувачів. Сучасні підходи передбачають використання багатофакторної автентифікації, одноразових паролів, перевірки електронної пошти або біометричних методів. Поєднання цих механізмів знижує ризик несанкціонованого доступу, навіть якщо один із факторів скомпрометовано.

1.3. Архітектура та вимоги до системи безпеки мобільного застосунку

Архітектура безпечного мобільного додатку повинна формуватися з урахуванням принципів безпеки за проектуванням та безпеки за замовчуванням, що передбачає інтеграцію механізмів захисту на ранніх етапах розробки. Такий підхід мінімізує ризики, пов'язані з подальшими змінами архітектури та усуненням вразливостей.

Система безпеки повинна забезпечувати конфіденційність, цілісність та доступність інформації. Для цього необхідно чітко розмежувати обов'язки між клієнтським додатком, серверною частиною та хмарною інфраструктурою. Клієнтський додаток повинен виконувати мінімально необхідну логіку та не зберігати критичні дані у відкритому доступі.

Для таких додатків, як онлайн-чати, особливо важливо забезпечити автентичність учасників взаємодії та захистити повідомлення від підміни. Це зумовлює необхідність використання сучасних криптографічних алгоритмів та безпечних каналів зв'язку.

1.4. Вибір інструментів та технологій

Вибір інструментів та технологій для реалізації безпечного мобільного застосунку базується на сучасних рекомендаціях щодо платформи Android та найкращих практиках розробки. У цій роботі як мова програмування було обрано Kotlin, яка забезпечує підвищену надійність коду, суворішу систему типів та зменшення кількості помилок під час виконання.

Для побудови інтерфейсу користувача було використано Jetpack Compose, який реалізує декларативний підхід до опису інтерфейсу користувача та спрощує управління станом. Як серверна інфраструктура було обрано платформу Firebase, яка забезпечує автентифікацію, зберігання даних та синхронізацію в режимі реального часу.

1.5. Модель загроз і сценарії атак

Модель загроз є фундаментальним елементом побудови системи безпеки. Вона дозволяє виявити потенційні джерела загроз, оцінити їх ймовірність та визначити можливі наслідки. Для мобільних додатків типові сценарії атак включають перехоплення мережевого трафіку, модифікацію клієнтського коду, несанкціонований доступ до локальних даних та компрометацію облікових записів. Для онлайн-чатів додаткові сценарії включають підміну повідомлень, повторне відтворення запитів та доступ до історії листування третіми особами. Аналіз цих сценаріїв дозволяє сформувати пріоритети для реалізації заходів захисту.

1.6. Розробка політик безпеки

Політики безпеки визначають набір правил та обмежень, спрямованих на забезпечення безпеки інформації в мобільному застосунку. Вони охоплюють питання доступу до даних, автентифікації користувачів, захисту мережевого з'єднання та реагування на інциденти безпеки.

Політики безпеки мають особливе значення для систем обміну повідомленнями, оскільки вони визначають правила обробки, зберігання та видалення повідомлень користувачів, а також відповідальність сторін у разі порушення безпеки.

2. РЕАЛІЗАЦІЯ БАГАТОРІВНЕВОГО ЗАХИСТУ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1. Створення прототипу мобільного застосунку (Android, Kotlin, Jetpack Compose)

В рамках практичної частини магістерської кваліфікаційної роботи було розроблено прототип мобільного додатку для безпечного обміну текстовими повідомленнями на платформі Android. Основною метою створення прототипу є перевірка ефективності багаторівневого підходу до забезпечення інформаційної безпеки в реальному програмному продукті, а також демонстрація практичного застосування сучасних технологій та механізмів захисту.

Прототип мобільного додатку було реалізовано з використанням мови програмування Kotlin, яка рекомендована Google для розробки Android-додатків. Kotlin забезпечує підвищену надійність коду завдяки суворій системі типів, підтримці null-безпеки та зменшенню кількості типових помилок виконання. Це важливий фактор у контексті безпеки, оскільки значна частина вразливостей у програмному забезпеченні виникає саме через помилки управління пам'яттю та неправильну обробку даних.

Для побудови інтерфейсу користувача було використано Jetpack Compose — сучасний декларативний фреймворк для створення інтерфейсу користувача в Android. Декларативний підхід дозволяє чітко відокремити логіку представлення від бізнес-логіки та спрощує управління станом інтерфейсу. Це знижує ризик логічних помилок, пов'язаних з невідповідними станами екрана, що особливо важливо для додатків з динамічною взаємодією в реальному часі.

Архітектура прототипу побудована з використанням шаблону MVVM (Model–View–ViewModel), який забезпечує розподіл відповідальності між компонентами програми. View відповідає за відображення даних та взаємодію з користувачем, ViewModel відповідає за логіку обробки та управління станом, а Model відповідає за роботу з даними та взаємодію з серверною частиною. Такий підхід дозволяє локалізувати критичну логіку безпеки у ViewModel та репозиторіях, мінімізуючи ризик її витоку або неправильного використання.

Функціонально прототип реалізує основні можливості системи онлайн-чату, включаючи:

- реєстрацію та автентифікацію користувачів;
- створення приватних діалогів між двома користувачами;
- обмін текстовими повідомленнями в режимі реального часу;
- перегляд історії повідомлень;
- автоматичну синхронізацію статусу чату між пристроями.

Хмарна платформа Firebase використовується для серверної та мережевої частин програми. Firebase Authentication відповідає за управління обліковими записами користувачів та перевірку їхньої автентичності, а Cloud Firestore використовується як база даних для зберігання службової інформації для чатів та повідомлень. Вибір Firebase зумовлений його підтримкою синхронізації даних у режимі реального часу, вбудованими механізмами безпеки та можливістю точного налаштування доступу до даних за допомогою правил безпеки.

Особливістю реалізованого прототипу є те, що всі повідомлення проходять криптографічну обробку на стороні клієнта перед збереженням у хмарному сховищі. Це означає, що серверна частина програми не має доступу до відкритого тексту повідомлень, а зберігає лише зашифровані дані та метадані служби. Такий підхід відповідає концепції наскрізного шифрування та значно підвищує рівень конфіденційності даних користувача.

Криптографічні ключі, необхідні для шифрування та розшифрування повідомлень, генеруються та зберігаються за допомогою системного механізму Android Keystore. Це дозволяє ізолювати закриті ключі від файлової системи та зменшити ризик їх компрометації навіть у разі доступу зловмисника до пристрою користувача. Відкриті ключі користувача зберігаються у хмарному сховищі та використовуються для узгодження симетричних ключів між сторонами обміну. У прототипі також реалізовано механізми контролю доступу до чатів та повідомлень. Кожен чат прив'язаний до певних учасників, а доступ до даних обмежений правилами безпеки Cloud Firestore. Це унеможливлює для користувачів, які не є учасниками відповідного діалогу, читання або зміну повідомлень, навіть у разі спроби виконання несанкціонованих запитів до бази даних.

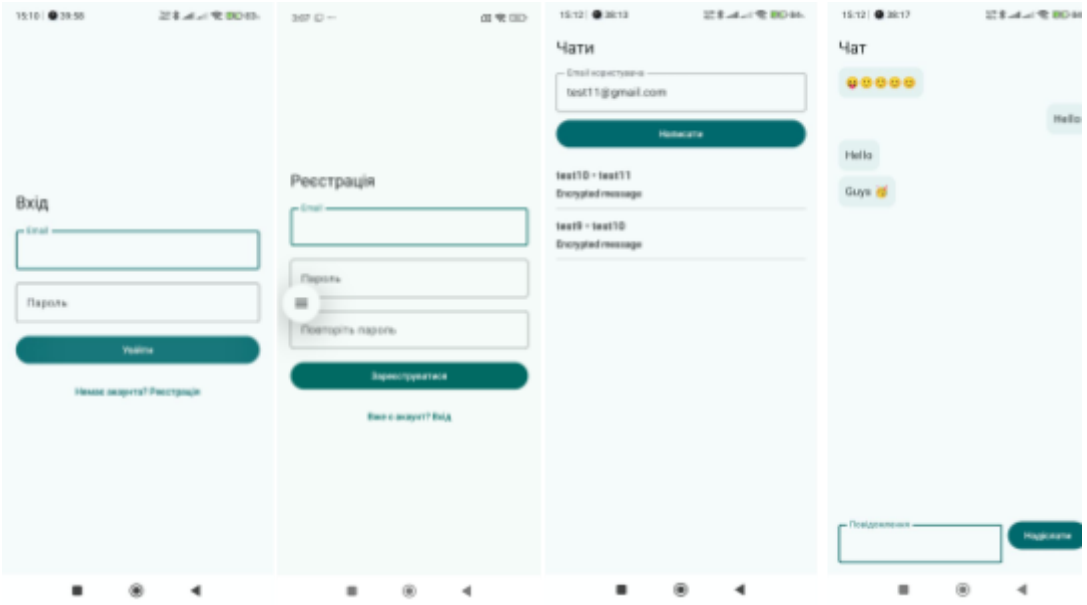

Рисунок 2.1 – UI основних екранів застосунку (авторизація, список чатів, екран відкритого чату)

Таким чином, створений прототип мобільного додатку є повноцінною експериментальною платформою для впровадження та тестування багаторівневих механізмів захисту. Він поєднує сучасні підходи до розробки Android-додатків з комплексними заходами інформаційної безпеки, що дозволяє використовувати його як основу для подальшого аналізу та вдосконалення системи захисту.

Однією з ключових загроз інформаційній безпеці мобільних Android-додатків є можливість зворотного проектування клієнтського коду та його подальшої модифікації. Через відкритий характер платформи Android, зловмисник може отримати APK-файл додатку, декомпілювати його, проаналізувати внутрішню логіку роботи та, за необхідності, змінити поведінку додатку, щоб обійти механізми захисту. У зв'язку з цим захист клієнтського коду є обов'язковим елементом багаторівневої системи безпеки.

У розробленому прототипі мобільного додатку для зменшення ризиків зворотного проектування були використані механізми обфускації та оптимізації коду, що надаються інструментом R8 (ProGuard), інтегрованим у стандартний ланцюжок збірки Android-додатків. Обфускація передбачає перейменування класів, методів та змінних на нетрадиційні назви, видалення невикористаного коду та оптимізацію байт-коду. В результаті, статичний аналіз програми та відновлення її початкової архітектури значно ускладнюються.

Додатково, у проекті використовується механізм shrinkResources, який дозволяє автоматично видаляти невикористовувані ресурси (зображення, рядки, стилі) з кінцевого APK-файлу. Це не тільки зменшує розмір програми, але й зменшує поверхню атаки, обмежуючи кількість елементів, які можуть бути використані зловмисником для аналізу або маніпуляцій.

Конфігурація підписання програми відіграє важливу роль у захисті від модифікації APK. Усі збірки програми підписуються цифровим сертифікатом, що гарантує цілісність та автентичність клієнтського коду. Будь-яка зміна APK-файлу після підписання призводить до втрати дійсності цифрового підпису, що унеможливлює встановлення або коректну роботу модифікованої версії програми на пристроях користувачів.

Окрім практично реалізованих механізмів, у контексті підвищення рівня захисту клієнтського коду доцільно використовувати сервіс Play Integrity API. Такий підхід дозволяє перевірити, чи працює програма на сертифікованому пристрої, чи був APK-файл модифікований, а також чи використовується емулятор або середовище з підвищеним рівнем ризику. Хоча пряма інтеграція Play Integrity API в рамках цього прототипу не

проводилася, його використання вважається перспективним напрямком подальшого розвитку системи безпеки.

Ще одним теоретичним напрямком підвищення захисту є реалізація механізмів виявлення втручання – виявлення змін у коді або середовищі виконання програми. Ці механізми включають перевірку контрольних сум, аналіз цілісності класів та виявлення ознак налагодження або інструментів динамічного аналізу. У поєднанні з перевірками на стороні сервера це дозволяє швидко блокувати доступ до сервісів у разі підозрілої поведінки клієнта.

Критично важливим аспектом захисту від модифікації клієнтського коду є перенесення ключових перевірок безпеки на сторону сервера. У розробленому додатку всі операції доступу до даних контролюються правилами безпеки Firebase Firestore, які перевіряють автентичність користувача та його приналежність до відповідного чату незалежно від стану клієнтського додатку. Таким чином, навіть у разі успішної модифікації APK, зловмисник не може обійти обмеження доступу до сервера.

Загалом, реалізований підхід до захисту клієнтського коду поєднує практичні механізми обфускації та оптимізації, коректну конфігурацію підписання, а також концептуальне обґрунтування використання сучасних сервісів перевірки цілісності та валідації сервера. У сукупності ці заходи значно ускладнюють зворотне проектування застосунку, зменшують ризик його модифікації та підвищують загальний рівень інформаційної безпеки системи.

Посилання на відкритий вихідний код проекту можна знайти у **Додатку А**.

2.2. Реалізація захисту коду (обфускація, перевірка цілісності)

Захист клієнтського коду є одним із базових і водночас критично важливих рівнів багаторівневої системи безпеки мобільного застосунку. На відміну від серверної частини, клієнтський код Android-застосунку фізично знаходиться на пристрої користувача, що робить його доступним для аналізу, модифікації та зворотного проєктування. У зв’язку з цим забезпечення стійкості клієнтського коду до реверс-інжинірингу є необхідною умовою побудови захищеного мобільного застосунку.

Однією з найбільш поширених загроз для Android-застосунків є **реверс-інжиніринг**, який передбачає аналіз байткоду з метою відновлення логіки роботи застосунку, пошуку вразливостей або отримання доступу до прихованих API та внутрішніх алгоритмів. За відсутності належного захисту зловмисник може отримати уявлення про структуру застосунку, механізми

автентифікації, обробку мережевих запитів або криптографічні операції, що значно спрощує подальші атаки.

Для ускладнення аналізу клієнтського коду у розробленому прототипі застосовано обфускацію, реалізовану засобами R8 (ProGuard), які є стандартною частиною інструментарію Android. Обфускація передбачає автоматичне перейменування класів, методів і змінних у неконвенційні імена, видалення невикористовуваного коду та оптимізацію структури байткоду. У результаті цього отриманий виконуваний файл втрачає логічну читабельність і стає значно складнішим для аналізу.

Використання R8 дозволяє також зменшити розмір застосунку та оптимізувати виконання, що позитивно впливає на продуктивність. Водночас основною метою застосування обфускації в даній роботі є саме підвищення рівня безпеки шляхом зменшення доступності внутрішньої логіки застосунку для сторонніх осіб.

Особливу увагу під час налаштування обфускації було приділено сумісності з хмарними сервісами Firebase та бібліотеками Google Play Services. Значна частина цих бібліотек використовує механізми рефлексії та серіалізації, що потребує збереження певних класів і метаданих у незмінному вигляді. Для цього у конфігурації ProGuard було визначено правила збереження (keep rules), які забезпечують коректну роботу механізмів автентифікації, доступу до бази даних і криптографічних операцій після обфускації.

Крім обфускації, важливим аспектом захисту клієнтського коду є **перевірка цілісності застосунку**. Цілісність означає, що виконуваний код не був змінений після інсталяції. У контексті мобільних застосунків порушення цілісності може свідчити про спроби модифікації коду, вбудовування шкідливих компонентів або підміну логіки виконання.

Хоча платформа Android забезпечує базові механізми перевірки підпису застосунку під час інсталяції та оновлення, цього часто недостатньо для протидії складним атакам. Тому архітектура розробленого прототипу побудована таким чином, щоб мінімізувати залежність безпеки від клієнтського коду. Критичні перевірки доступу та авторизації виконуються на рівні серверної інфраструктури за допомогою правил безпеки Cloud Firestore, що унеможливлює обхід захисту шляхом модифікації клієнта.

Додатковим елементом захисту є **мінімізація критичної логіки на стороні клієнта**. Усі операції, які можуть впливати на безпеку даних, зокрема перевірка прав доступу до чатів і повідомлень, не покладаються виключно на локальні перевірки. Навіть у разі модифікації клієнтського застосунку зловмисник не зможе отримати доступ до даних, якщо серверна частина відхиляє запити, що не відповідають правилам безпеки.

Також у процесі розробки мобільного застосунку було дотримано принципу least privilege, відповідно до якого клієнтський застосунок отримує доступ лише до мінімально необхідних системних ресурсів і дозволів. Реалізація цього принципу дозволяє зменшити потенційні наслідки компрометації клієнтської частини та обмежити можливості зловмисника у разі успішної атаки.

Для підвищення стійкості клієнтського коду до реверс-інжинірингу та несанкціонованої модифікації застосовано механізми обфускації та оптимізації байткоду з використанням інструменту R8 (ProGuard), вбудованого в інструментарій платформи Android. Обфускація передбачає перейменування класів, методів і змінних, видалення невикористовуваного коду та ускладнення структури виконуваного файлу, що суттєво ускладнює статичний аналіз застосунку.

З метою забезпечення коректної роботи хмарних сервісів Firebase та Google Play Services у процесі обфускації налаштовано спеціальні правила збереження критичних класів і метаданих. Зокрема, у конфігурації ProGuard передбачено збереження класів, що використовуються для автентифікації, обробки асинхронних задач та серіалізації даних, а також анотацій і метаданих мови Kotlin, необхідних для коректної роботи механізмів рефлексії.

Окрему увагу приділено збереженню класів моделей даних, які використовуються для взаємодії з хмарною базою даних Firebase Firestore. Для цього у конфігурації ProGuard визначено правила, що запобігають обфускації відповідних класів і їхніх полів, що гарантує коректну серіалізацію та десеріалізацію даних під час виконання застосунку.

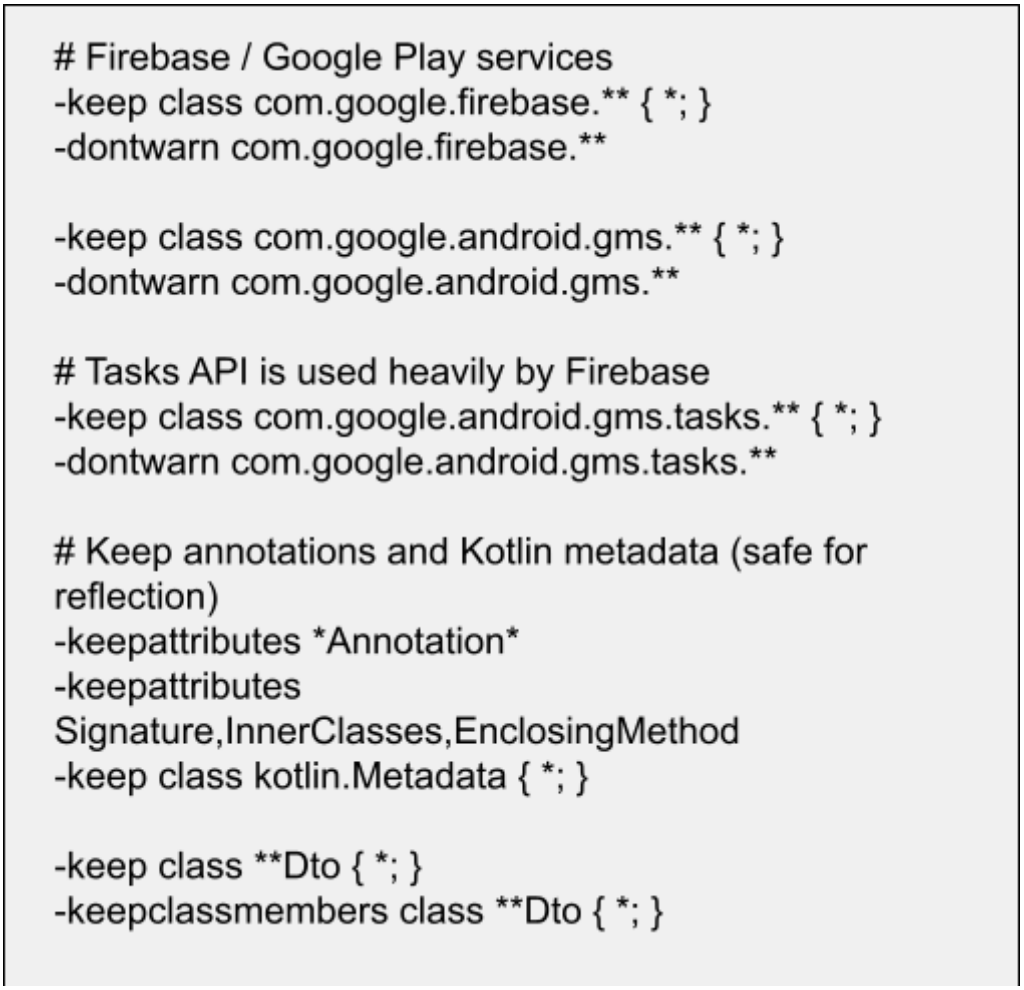

Рисунок 2.2 – proguard налаштування проекту

Таким чином, реалізований підхід до захисту клієнтського коду поєднує застосування принципу найменших привілеїв, обфускацію та оптимізацію коду, а також мінімізацію критичної логіки на стороні клієнта зі суворим контролем доступу на рівні сервера. У сукупності ці заходи значно ускладнюють зворотне проектування, модифікацію клієнтського застосунку та спроби обійти реалізовані механізми безпеки.

Додаткові аспекти захисту клієнтського коду мобільного застосунку. Окрім використання механізмів обфускації, важливим аспектом захисту клієнтського коду є зменшення кількості критичної бізнес-логіки, яка виконується безпосередньо на стороні мобільного застосунку. У розробленому прототипі логіка, пов'язана з контролем доступу до даних, автентифікацією користувачів та перевіркою прав доступу, максимально перенесена на рівень сервера та реалізована за допомогою правил безпеки Firebase Firestore. Такий підхід дозволяє зменшити ризики, пов'язані з модифікацією клієнтського застосунку або його виконанням у зміненому середовищі.

Важливим компонентом захисту коду також є обмеження використання механізмів динамічного виконання та завантаження коду. В рамках розробленого застосунку не використовується динамічне завантаження виконуваних модулів, відображення або виконання коду із зовнішніх джерел, що значно зменшує поверхню атаки та відповідає рекомендаціям OWASP щодо уникнення небезпечних практик роботи з кодом на мобільних платформах.

Для ускладнення статичного та динамічного аналізу застосунку також було використано підхід з чітким розділенням архітектурних шарів. Компоненти інтерфейсу користувача, логіки станів та доступу до даних реалізовані в окремих модулях з чітко визначеними інтерфейсами взаємодії. Це ускладнює розуміння загальної логіки роботи застосунку у випадку аналізу декомпільованого коду та зменшує ймовірність швидкого відтворення внутрішньої архітектури.

Особлива увага приділяється захисту конфігураційної інформації та запобіганню жорсткому кодуванню конфіденційних значень. У клієнтському коді немає жорстко закодованих криптографічних ключів, токенів доступу або секретів служб. Усі конфіденційні параметри або генеруються динамічно, або зберігаються в захищених системних механізмах платформи Android. Це зменшує ризик витоку даних під час аналізу APK-файлу та відповідає рекомендаціям щодо безпечного зберігання секретів у мобільних застосунках.

Таким чином, захист клієнтського коду в розробленому мобільному додатку реалізовано як комплекс взаємодоповнюючих заходів, що включають обфускацію, архітектурну ізоляцію компонентів, мінімізацію критичної логіки на стороні клієнта та дотримання принципів безпечної розробки. У поєднанні з контролем доступу на стороні сервера та криптографічним захистом даних ці заходи значно ускладнюють зворотну інженерію та знижують ризик компрометації механізмів безпеки мобільного додатку.

2.3. Реалізація захисту даних (Keystore, шифрування, біометрія)

Захист даних користувачів є одним із ключових аспектів безпеки мобільних додатків, особливо у випадку систем обміну повідомленнями, які обробляють конфіденційну інформацію. У таких додатках порушення конфіденційності або цілісності даних може призвести до серйозних наслідків, включаючи витік особистої інформації, компрометацію приватного місцезнаходження та порушення довіри користувача до системи. У зв'язку з цим розроблений прототип реалізує комплексний підхід до захисту даних, який включає криптографічне шифрування, безпечне зберігання ключів та використання механізмів системної безпеки платформи Android.

Загальні принципи захисту даних у мобільних застосунках

Основні вимоги до захисту даних у мобільних додатках полягають у забезпеченні конфіденційності, цілісності та доступності інформації. Конфіденційність передбачає неможливість ознайомлення з даними третіх осіб, цілісність – захист від несанкціонованої модифікації, а доступність – гарантований доступ до даних для авторизованих користувачів. У випадку мобільних чатів особливе значення має наскрізне шифрування, при якому повідомлення шифруються на пристрої відправника та можуть бути розшифровані лише на пристрої одержувача. Серверна частина в такій архітектурі виконує лише функцію транспортування зашифрованих даних і не має доступу до їх вмісту.

Використання Android Keystore для зберігання криптографічних ключів

Для безпечного зберігання криптографічних ключів розроблений застосунок використовує механізм Android Keystore, який є системним компонентом операційної системи Android. Android Keystore дозволяє створювати та зберігати криптографічні ключі в безпечному середовищі, ізольованому від файлової системи застосунку.

Ключовою перевагою Android Keystore є те, що закриті ключі неможливо експортувати з пристрою у відкритому тексті. Навіть якщо зловмисник отримає доступ до файлової системи або пам'яті застосунку, криптографічні ключі залишаються недоступними. У багатьох сучасних пристроях Keystore інтегровано з апаратними модулями безпеки (Trusted Execution Environment або Secure Enclave), що ще більше підвищує рівень захисту.

У рамках прототипу для кожного користувача під час першої автентифікації генерується асиметрична пара криптографічних ключів. Закритий ключ зберігається виключно в Android Keystore, а відкритий ключ публікується в хмарному сховищі Firebase Firestore та використовується іншими учасниками чату для встановлення безпечного каналу обміну повідомленнями.

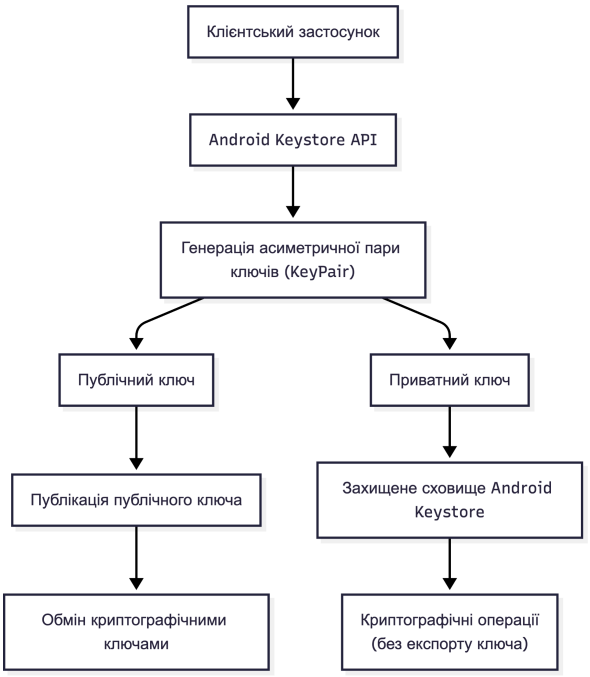

Рисунок 2.3 – Схема генерації та зберігання криптографічних ключів із використанням Android Keystore

Криптографічний обмін ключами та формування симетричного ключа

Для реалізації наскрізного шифрування застосунок використовує схему, засновану на алгоритмі еліптичної кривої Діффі-Хеллмана (ECDH). Цей алгоритм дозволяє двом сторонам, які мають власні асиметричні пари ключів, узгодити спільний секретний ключ без необхідності передачі його через мережу.

Процес обміну ключами в розробленому застосунку виглядає наступним чином:

- Кожен користувач має свою власну асиметричну пару ключів.
- Відкритий ключ користувача зберігається у Firebase Firestore.
- Під час надсилання повідомлення відправник отримує відкритий ключ одержувача.
- На основі власного закритого ключа та відкритого ключа співрозмовника виконується розрахунок спільного секрету за допомогою ECDH.
- Отриманий секрет використовується для формування симетричного ключа шифрування.

Для підвищення криптографічної стійкості додатково використовується значення солі, сформоване на основі ідентифікатора чату. Це дозволяє уникнути повторного використання одного й того ж ключа в різних чатах.

Симетричне шифрування повідомлень

Симетричний ключ, отриманий в результаті обміну ECDH, використовується для шифрування текстових повідомлень перед їх відправкою до хмарного сховища. Для цього використовується сучасний алгоритм симетричного шифрування, який забезпечує високий рівень криптографічної стійкості та ефективності виконання на мобільних пристроях.

Шифрування виконується локально на пристрої користувача перед відправкою повідомлення. Firebase Firestore зберігає лише зашифровані дані разом із службовою інформацією, необхідною для подальшого розшифрування (наприклад, вектор ініціалізації). Таким чином, навіть якщо серверна інфраструктура буде скомпрометована, зловмисник не зможе отримати доступ до вмісту повідомлень.

Під час отримання повідомлення клієнтська програма виконує зворотну операцію – формує симетричний ключ на основі власного закритого ключа та відкритого ключа співрозмовника, а потім розшифровує повідомлення в пам'яті пристрою.

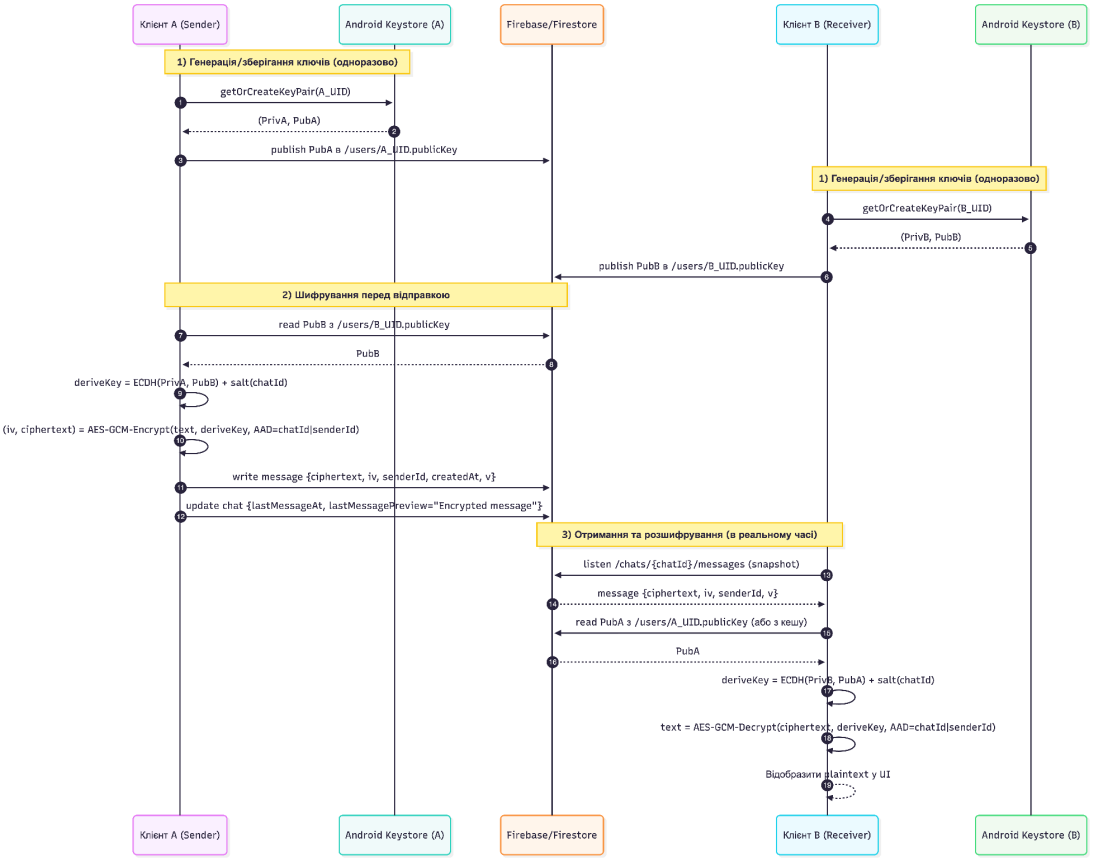

Рисунок 2.4 – Діаграма процесу шифрування та розшифрування повідомлень у мобільному Android-застосунку

Захист даних у стані спокою

Окрім захисту даних під час передачі, важливим аспектом є захист інформації в стані спокою, тобто під час зберігання на пристрої або в хмарі. Розроблений застосунок не зберігає відкритий текст повідомлень у локальному сховищі пристрою. Дані, що кешуються для забезпечення коректної роботи інтерфейсу, зберігаються в зашифрованому вигляді або лише в пам'яті застосунку.

Хмарне сховище Firebase Firestore зберігає лише зашифровані повідомлення, ідентифікатори користувачів та службову інформацію. Доступ до цих даних суворо обмежений правилами безпеки Firestore, які дозволяють читання та запис лише учасникам відповідного чату.

Використання біометрії як додаткового рівня захисту

Для підвищення рівня безпеки локального доступу до застосунку, архітектура прототипу передбачає можливість використання біометричної автентифікації, зокрема, розпізнавання відбитків пальців або обличчя. Біометрія може бути використана як додатковий рівень захисту під час відкриття застосунку або доступу до історії повідомлень. Інтеграція біометричної автентифікації базується на стандартних API платформи Android, що забезпечує сумісність з різними пристроями та дотримання рекомендацій щодо безпеки. Використання біометрії не замінює основні механізми автентифікації, а доповнює їх, знижуючи ризик несанкціонованого доступу у разі фізичного доступу до пристрою.

Узагальнення реалізованого підходу

Таким чином, у розробленому прототипі реалізовано багаторівневий захист даних, який внаслідок:

- безпечне зберігання криптографічних ключів за допомогою Android Keystore;
- наскрізне шифрування повідомляє з використанням асиметричних і симетричних алгоритмів;
- захист даних у стані передачі та у стані спокою;
- можливість використання біометричної автентифікації як додаткового рівня захисту.

Реалізований підхід відповідає сучасним вимогам інформаційної безпеки та рекомендаціям OWASP Mobile Application Security Verification

Standard і може бути використаний як основа для створення захищених мобільних систем обміну повідомленнями.

Одним із ключових аспектів багаторівневого захисту мобільного застосування є забезпечення конфіденційності та цілісності даних користувачів. У контексті системи обміну повідомленнями це має особливе значення, такі застосунки працюють з персональною інформацією, яка передається та зберігається у найбільшій кількості. У розробленому прототипі реалізовано комплексний підхід до захисту даних, що компенсується використанням системних механізмів Android та сучасних криптографічних алгоритмів.

Основою захисту криптографічних ключів у застосуванні є використання Android Keystore — системного сховища, яке дозволяє генерувати та зберігати криптографічні ключі в захищеному середовищі пристрою. Приватні ключі створюються в центрі Keystore і не можуть бути експортованими або прочитаними клієнтським кодом. Криптографічні операції, зокрема підписання та обчислення спільних секретів, використовуються в захищеному середовищі, що значно знижує ризик компрометації ключового матеріалу навіть у разі отримання зловмисником доступу до пристрою файлової системи.

Для реалізації нарізного шифрування повідомляється застосована схема обміну ключами на основі алгоритму ECDH (Elliptic Curve Diffie–Hellman). Кожен користувач має власну асиметричну пару ключів, де публічний ключ зберігається в хмарній базі даних та використовується іншими учасниками чату для встановлення спільного симетричного ключа. Отриманий симетричний ключ використовується для шифрування повідомлень за допомогою алгоритму AES у режимі GCM, що забезпечує одночасно конфіденційність і цілість даних.

Застосування режиму AES-GCM дозволяє виявляти будь-які спроби модифікації зашифрованого повідомлення, перевірка автентичності даних виконується під час розшифрування. Усі повідомлення перед збереженням у Firebase Firestore зберігаються в зашифрованому вигляді, а серверна частина не має доступу до відкритого текстового повідомлення. Такий підхід відповідає принципам наскрізного шифрування та значному рівню захисту конфіденційної інформації.

Крім того, архітектура забезпечує можливість розширення механізмів захисту шляхом інтеграції біометричної аутентифікації для доступу до криптографічних операцій. Android Keystore підтримує обмеження використання ключів із застосуванням біометричних факторів, що дозволяє додатково захистити дані користувача у разі втрати або компрометації пристрою.

Ефективне управління криптографічними ключами є одним із ключових факторів забезпечення конфіденційності та цільності даних у мобільних застосунках. Використання сучасних криптографічних алгоритмів не гарантує належного рівня безпеки у випадку неналежного зберігання, обробки або розповсюдження криптографічних ключів. У зв'язку з цими ключами управління необхідно розглядати як окремий, повноцінний рівень системи інформаційної безпеки.

У розробленому мобільному застосунку реалізовано модель управління ключами, що обґрунтовується на принципі розмежування відповідальності між клієнтською та серверною частинами. Генерація та зберігання приватних криптографічних ключів виконується на стороні клієнта з використанням механізму Android Keystore. Це дозволяє гарантувати, що приватні ключі ніколи не залишають межі пристрою користувача та не передають мережу у відкритому або зашифрованому вигляді.

Android Keystore забезпечує апаратне або програмне збереження середовища збереження ключів, що унеможливлює їх пряме вилучення навіть у разі отримання доступу до пристрою файлової системи. У межах застосування для кожного користувача створена асиметрична пара ключів, яка використовується для реалізації механізму наскрізного шифрування повідомлень. Приватний ключ використовується тільки для виконання криптографічних операцій, тоді як публічний ключ може бути безпечно опублікований у хмарному сховищі.

Публічні ключі користувачів зберігаються в базі даних Firebase Firestore у складі профілю користувача та використовують для створення симетричних ключів між учасниками чату. Такий підхід дозволяє реалізувати обмін зашифрованими повідомленнями без необхідності централізованого управління секретними ключами на сервері, що відповідає концепції наскрізного шифрування.

Важливим аспектом управління криптографічними ключами є їх життєвий цикл, який включає етапи генерації, зберігання, використання та, у разі необхідності, ротації або відкликання. У межах розробленого прототипу генерація ключового пари виконується автоматично під час першої автентифікації користувача або перевіряється при кожному вході до системи. У разі слабких ключів або їх пошкодження застосунок ініціює повторну генерацію з подальшим оновленням публічного ключа у хмарному сховищі.

Ротація ключів, хоча й не реалізована в повному обсязі в даному прототипі, розглядається як важливий напрям подальшого розвитку системи. У практичних системах обміну повідомленнями ротація ключів дозволяє зменшити наслідки наявної компрометації та обмежити обсяги даних, які можуть бути розшифровані у разі витоку одного з ключів.

Окрему увагу приділено мінімізації доступу до криптографічних матеріалів відповідно до принципу найменших привілеїв. Клієнтський застосунок не має прямого доступу до приватного ключа у вигляді даних, а лише ініціює криптографічні операції через API Android Keystore. Це суттєво зменшує ризик витоку ключів у разі реверс-інжинірингу або динамічного аналізу застосування.

Таким чином, реалізована модель управління криптографічними ключами за рахунок збереження зберігання, чітко визначений життєвий цикл ключів та мінімізація доступу до критичних криптографічних матеріалів. У сукупності це забезпечує надійний фундамент для реалізації наскрізного шифрування та забезпечення загального рівня інформаційної безпеки мобільного застосування.

2.4. Реалізація безпечної автентифікації (MFA, OTP, біометрія)

Автентифікація користувачів становить один з основних аспектів безпеки мобільного застосунку, оскільки саме на цьому етапі вирішується питання доступу користувача до конфіденційної інформації та функціональних можливостей системи. Слабкі механізми автентифікації часто є причиною компрометації облікових записів у мобільних застосунках, про що свідчать статистичні дані щодо інцидентів безпеки та рекомендації OWASP Mobile Top 10.

У рамках створеного прототипу безпечного мобільного чату реалізовано багаторівневий підхід до автентифікації, який об'єднує традиційну автентифікацію за обліковими даними, підтвердження електронної пошти, додаткові фактори автентифікації та підтримку біометричних механізмів платформи Android.

Базова автентифікація за обліковими даними

Як основний метод ідентифікації користувачів у застосунку використовується автентифікація за допомогою електронної пошти та пароля, яка реалізується через сервіс Firebase Authentication. Цей підхід є одним із найбільш розповсюджених і має підтримку більшості користувачів, що забезпечує зручний та зрозумілий процес входу в систему.

Firebase Authentication виконує функції перевірки облікових даних, безпечного зберігання хешованих паролів та управління сесіями користувачів. Це дозволяє клієнтському застосунку уникати обробки та зберігання паролів у відкритому вигляді, тим самим знижуючи ризик їх компрометації в разі аналізу клієнтського коду або витоку локальних даних.

У створеному прототипі логіка автентифікації реалізована на рівні ViewModel, що відповідає принципам архітектури MVVM і дозволяє чітко відокремити бізнес-логіку від користувацького інтерфейсу. Інтерфейс застосунку реагує тільки на зміни стану автентифікації, що спрощує підтримку й удосконалення механізмів безпеки.

Підтвердження електронної пошти як елемент багатофакторної автентифікації

Одним із ключових аспектів підвищення безпеки облікових записів є верифікація електронної пошти користувача. У рамках прототипу було впроваджено обов'язкову підтвердження email-адреси після реєстрації нових користувачів.

Після створення облікового запису Firebase автоматично надсилає листа з посиланням для підтвердження електронної пошти. До моменту завершення цього процесу доступ до основних функцій застосунку є обмеженим. Такий підхід дозволяє:

- зменшити кількість фальшивих або автоматично створених облікових записів;
- підтвердити, що користувач дійсно володіє зазначеною електронною адресою;
- підвищити рівень довіри до ідентифікатора користувача.

З точки зору інформаційної безпеки, верифікація електронної пошти може розглядатися як додатковий фактор автентифікації, який доповнює традиційну комбінацію «логін–пароль».

Багатофакторна автентифікація (MFA) у контексті Firebase

Багатофакторна автентифікація (MFA) передбачає використання двох або більше незалежних елементів для підтвердження особи користувача. У мобільних застосунках такими елементами можуть бути:

- Знання (пароль, PIN-код);

- Володіння (пристрій, одноразовий код);

- Біометричні характеристики (відбиток пальця, обличчя).

Архітектура Firebase Authentication підтримує реалізацію MFA шляхом поєднання базової автентифікації з додатковими методами перевірки. У рамках даного прототипу MFA реалізовано через підтвердження електронної

пошти, що є практичним і універсальним рішенням для застосунків, призначених для широкого кола користувачів.

Цей підхід є особливо виправданим у контексті навчального та дослідницького проєкту, оскільки він не вимагає додаткових апаратних ресурсів або зовнішніх сервісів, але суттєво підвищує рівень захисту облікових записів.

Одноразові паролі та токени доступу

Під час автентифікації Firebase Authentication використовує механізм короткоживучих токенів доступу, які автоматично оновлюються та перевіряються серверною інфраструктурою. Хоча ці токени не є класичними одноразовими паролями, вони виконують подібну функцію — обмежують тривалість сесії і знижують ризик повторного використання скомпрометованих даних.

Клієнтський застосунок не має безпосереднього доступу до вмісту токенів і не здійснює їх ручне управління, що зменшує ймовірність помилок у реалізації. Усі запити до хмарних сервісів виконуються лише в межах дійсної сесії користувача, а контроль доступу додатково забезпечується правилами безпеки Firestore.

Використання біометричної автентифікації

У прототипі біометрична автентифікація розглядається як додатковий локальний рівень захисту, що може бути застосований після успішного входу користувача в систему. Біометричні дані не використовуються як основний метод ідентифікації; натомість вони слугують для захисту доступу до застосунку або окремих його функцій у разі фізичного доступу до пристрою.

Інтеграція біометрії ґрунтується на стандартних API Android, що забезпечує:

- Сумісність із різними типами біометричних сенсорів;
- Відповідність системним вимогам безпеки;
- Відсутність необхідності обробляти біометричні дані безпосередньо самими застосунками.

Запровадження біометричної автентифікації дозволяє знизити ризик несанкціонованого доступу до конфіденційної інформації в разі втрати або крадіжки мобільного пристрою.

Узагальнення реалізованого підходу до автентифікації

Отже, у розробленому мобільному застосунку впроваджено багаторівневу систему автентифікації, яка поєднує:

- Автентифікацію за електронною поштою та паролем;

- Обов’язкове підтвердження електронної пошти;

- Використання захищених сесій та токенів доступу;

- Можливість впровадження біометричної автентифікації.

Цей підхід відповідає сучасним вимогам інформаційної безпеки та рекомендаціям OWASP Mobile Application Security Verification Standard і забезпечує оптимальний баланс між зручністю для користувача і високим рівнем захисту облікових записів.

2.5. Реалізація захисту мережевих з’єднань (TLS, сертифікат пінінг)

Мережеві з’єднання є одним з найбільш уразливих компонентів мобільних застосунків, оскільки саме через них здійснюється передача конфіденційної інформації між клієнтською та серверною частинами системи. Для мобільних застосунків типу онлайн-чатів загрози, пов’язані з мережевою взаємодією, включають перехоплення трафіку, атаки типу «людина посередині» (Man-in-the-Middle), підміну серверів, повторне використання мережевих сесій та аналіз зашифрованих потоків даних. У зв’язку з цим захист мережевих з’єднань є обов’язковим елементом багаторівневої системи безпеки.

Загальні принципи захисту мережевих з’єднань у мобільних застосунках. Основним принципом безпечної мережевої взаємодії є використання криптографічно захищених каналів зв’язку, які забезпечують конфіденційність, цілісність та автентичність переданих даних. У сучасних мобільних застосунках таким стандартом є використання протоколу Transport Layer Security (TLS), який реалізує захищене з’єднання поверх транспортного рівня мережі.

Протокол TLS забезпечує:

- шифрування переданих даних;

- перевірку автентичності сервера за допомогою цифрових сертифікатів;

- захист від підміни або модифікації даних під час передачі.

Використання TLS є обов’язковою вимогою платформи Android для більшості мережевих взаємодій, а починаючи з новіших версій операційної системи незахищені HTTP-з’єднання за замовчуванням блокуються.

Захист мережевих з'єднань у розробленому прототипі. У межах розробленого мобільного застосунку всі мережеві взаємодії здійснюються виключно через хмарну платформу Firebase, яка за замовчуванням використовує захищені TLS-з'єднання. Це стосується як процесів автентифікації користувачів, так і обміну даними з Firestore у режимі реального часу.

Завдяки використанню Firebase клієнтський застосунок не взаємодіє безпосередньо з відкритими HTTP-ендпойнтами. Усі запити виконуються через офіційні SDK, які реалізують сучасні криптографічні алгоритми та автоматично перевіряють дійсність сертифікатів серверів. Таким чином, базовий рівень захисту мережевих з'єднань забезпечується на рівні платформи та не залежить від помилок реалізації на стороні розробника.

Сертифікатне пінінгування як додатковий рівень захисту. Хоча стандартна перевірка сертифікатів у межах TLS забезпечує високий рівень безпеки, у деяких сценаріях можливі атаки, пов'язані з компрометацією центрів сертифікації або встановленням на пристрій користувача шкідливих кореневих сертифікатів. У таких випадках навіть TLS-з'єднання може бути перехоплене.

Для зменшення ризику подібних атак у безпечних мобільних застосунках застосовується механізм сертифікатного пінінгування. Суть цього підходу полягає у жорсткому зв'язуванні клієнтського застосунку з конкретним сертифікатом або публічним ключем сервера. Якщо під час встановлення з'єднання сертифікат не відповідає очікуваному, з'єднання відхиляється незалежно від його формальної дійсності.

У контексті розробленого прототипу сертифікатне пінінгування розглядається як додатковий рівень захисту, який може бути реалізований у разі використання власних мережевих API або кастомних серверів. Водночас при роботі з Firebase повна реалізація пінінгування у клієнтському застосунку є обмеженою, оскільки керування сертифікатами серверної інфраструктури здійснюється Google. Проте сам факт використання офіційної хмарної платформи з високими стандартами безпеки суттєво знижує ризики мережевих атак.

Поєднання мережевого захисту з клієнтським шифруванням. Важливою особливістю реалізованої архітектури є те, що захист мережевих з'єднань не розглядається як єдиний або достатній механізм забезпечення конфіденційності. Навіть у разі гіпотетичного компрометування TLS-з'єднання зміст переданих повідомлень залишатиметься недоступним для зловмисника, оскільки дані додатково зашифровані на стороні клієнта.

Таким чином, у системі реалізовано принцип defense in depth, за якого мережевий захист доповнюється криптографічним захистом даних на прикладному рівні. Це дозволяє мінімізувати наслідки окремих успішних атак і забезпечити стійкість системи навіть у несприятливих умовах.

Захист від атак повторного використання та несанкціонованого доступу. Окрім шифрування з'єднань, важливим аспектом мережевої безпеки є захист від повторного використання сесій та несанкціонованих запитів. Firebase Authentication використовує короткоживучі токени доступу, які автоматично оновлюються та перевіряються серверною частиною. Це унеможливлює тривале використання скомпрометованих токенів та зменшує ризик атак типу replay.

Додатково правила безпеки Firestore виконують перевірку автентичності користувача та його прав доступу на кожен запит, незалежно від того, з якого клієнта він був ініційований. Таким чином, навіть у разі перехоплення мережевого трафіку або підміни клієнта зловмисник не зможе отримати доступ до даних без дійсної автентифікації.

Узагальнення реалізованого підходу до захисту мережевих з'єднань. У межах розробленого мобільного застосунку реалізовано комплексний підхід до захисту мережевих з'єднань, який включає:

- використання захищених TLS-з'єднань для всіх мережевих взаємодій;
- перевірку автентичності серверів за допомогою цифрових сертифікатів;
- можливість застосування сертифікатного пінінгування у разі використання власних API;
- поєднання мережевого захисту з клієнтським наскрізним шифруванням повідомлень;
- використання захищених токенів доступу та серверних правил контролю доступу.

Завдяки цьому мережевий рівень безпеки органічно доповнює інші компоненти багаторівневої системи захисту та забезпечує високий рівень стійкості мобільного застосунку до сучасних мережевих загроз.

Платформа Firebase автоматично забезпечує використання сучасних версій протоколу TLS та надійних криптографічних алгоритмів, що зменшує

ризик атак типу «людина посередині». Однак лише захищеного транспортного рівня недостатньо для забезпечення повної конфіденційності даних, оскільки серверна частина все ще має доступ до переданого вмісту.

Саме тому у розробленому застосунку захист мережевих з’єднань доповнено клієнтським наскрізним шифруванням повідомлень. Навіть у разі компрометації мережевого каналу або серверної інфраструктури зловмисник не може отримати доступ до відкритого тексту повідомлень без відповідних криптографічних ключів. Такий підхід дозволяє розділити відповідальність між транспортним та прикладним рівнями безпеки.

Крім того, архітектура застосунку мінімізує обсяг переданих службових даних і не використовує небезпечні механізми міжпроцесної взаємодії або неявні інтент-фільтри для обміну інформацією. Усі мережеві операції виконуються лише після успішної автентифікації користувача, що зменшує ризик несанкціонованого доступу до серверних ресурсів.

У сукупності використання захищених мережевих протоколів, клієнтського шифрування та строгих правил доступу на серверному рівні забезпечує високий рівень захисту даних під час їх передачі та відповідає сучасним рекомендаціям у сфері кібербезпеки мобільних застосунків.

3. ПРОПОЗИЦІЇ ЩОДО МОЖЛИВИХ ШЛЯХІВ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ

3.1. Методика тестування системи безпеки

Тестування системи безпеки мобільного додатку є невід'ємною частиною процесу його розробки та впровадження. Крім того, за умови правильної реалізації захисних механізмів на етапі проектування, демонстрація тестування системи може призвести до наявності прихованих вразливостей, які можуть бути використані зловмисниками. В рамках цієї магістерської роботи розроблено та застосовано методологію тестування системи безпеки мобільного месенджера Android, яка дозволяє комплексно оцінити ефективність впроваджених заходів захисту.

Запропонована методологія тестування базується на комбінованому функціональному, інтеграційному та безпековому тестуванні та відповідає рекомендаціям Стандарту перевірки безпеки мобільних додатків OWASP (MASVS).

Загальні принципи методології тестування. Методологія тестування системи безпеки мобільного додатку базується на таких принципах:

- систематичність - перевірка всіх рівнів безпеки додатку;
- відтворюваність - можливість повторення тестів;
- близькість до реальних умов експлуатації;
- орієнтація на ілюстративні сценарії атак.

Тестування з точки зору зловмисника, який має доступ до клієнтського додатку, але не має прав адміністратора в серверній інфраструктурі. Це відповідає найсучаснішій моделі загроз для мобільних додатків.

Функціональне тестування механізмів безпеки. Функціональне тестування спрямоване на перевірку правильності роботи реальних механізмів безпеки у стандартних сценаріях використання. На цьому етапі було протестовано такі аспекти:

- коректність процесу автентифікації та повторного входу користувача;
- обмеження доступу до функціональності додатку для неавторизованих користувачів;
- коректність роботи підтвердження електронної пошти;
- поведінка додатку у разі втрати сеансу або закінчення терміну дії токенів доступу.

Функціональне тестування підтвердило, що користувач не має доступу до списку чату та звітів без проходження повного процесу автентифікації. У разі завершення сеансу або виходу з професійного облікового запису всі захищені ресурси залишаються недоступними.

Інтеграційне тестування взаємодії клієнт-сервер. Інтеграційне тестування спрямоване на перевірку взаємодії клієнтського додатку з хмарною інфраструктурою Firebase. Особлива увага була приділена перевірці правильності правил безпеки Firestore та обмежень доступу до даних.

У рамках цього етапу було проведено такі перевірки:

- спроби читання документів чату користувачем, який не є учасником;
- спроби створення повідомлень від імені іншого користувача;
- спроби доступу до колекції без дійсної автентифікації;
- перевірка ізоляції даних між швидкими чатами.

Для тестування використовувалися як реальні клієнтські сесії, так і модифіковані запити, ініційовані інструментами емуляції. Результати тестування показали, що всі спроби несанкціонованого доступу були відхилені на рівні сервера відповідно до визначених правил безпеки.

Важливим елементом багаторівневого захисту мобільних додатків є контроль доступу до даних на рівні сервера. Розроблений прототип використовує для цього механізм правил безпеки Firebase Firestore, який дозволяє обмежувати операції читання та запису даних залежно від статусу автентифікації користувачів та ролей у системі.

Правила безпеки Firestore повинні залишатися на сервері та не залежати від коректності реалізації клієнтського додатку, яка не може бути обмежена обмеженнями навіть у разі модифікації або компрометації клієнтського коду. Таким чином, контроль доступу на стороні сервера є критично рівноцінним захистом, що доповнює механізми автентифікації та шифрування клієнта.

На рисунку показано фрагмент реалізованих правил безпеки Firebase Firestore, які регулюють доступ до колекції користувачів, чатів та повідомлень.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    function signedIn() {
      return request.auth != null;
    }

    match /users/{uid} {
      allow read: if signedIn();
      allow create, update, delete: if signedIn() && request.auth.uid == uid;
    }

    match /chats/{chatId} {

      allow create: if signedIn()
        && request.resource.data.participants is list
        && request.auth.uid in request.resource.data.participants;

      allow read: if signedIn()
        && request.auth.uid in resource.data.participants;

      allow update, delete: if signedIn()
        && request.auth.uid in resource.data.participants
        && request.resource.data.participants == resource.data.participants;

      match /messages/{messageId} {

        allow read: if signedIn()
          && request.auth.uid in
            get(/databases/$(database)/documents/chats/$(chatId)).data.participants;

        allow create: if signedIn()
          && request.resource.data.senderId == request.auth.uid
          && request.auth.uid in
            get(/databases/$(database)/documents/chats/$(chatId)).data.participants;

        allow update, delete: if false;
      }
    }
  }
}
```

Рисунок 3.1 – Правила безпеки Firestore

Тестування захисту даних та наскрізного шифрування. Окремим етапом методології є тестування реалізованого механізму наскрізного шифрування повідомлень. Метою цього етапу є перевірка того, що конфіденційні дані не зберігаються та не передаються у відкритому вигляді.

Тестування включало:

- аналіз вмісту документів у Firestore за допомогою консолі адміністратора;
- перевірку неможливості читання повідомлень без закритого криптографічного ключа;
- перевірку коректності розшифрування повідомлень лише на пристроях учасників чату;
- аналіз поведінки програми у разі зміни або втрати криптографічних ключів.

Результати тестування підтвердили, що у хмарному сховищі зберігаються лише зашифровані повідомлення, а спроби інтерпретувати їх вміст без відповідних ключів не дозволяють отримати відкритий текст.

Тестування стійкості клієнтського коду до модифікації. З метою оцінки ефективності захисту клієнтського коду було проведено тестування стійкості програми до модифікації. Зокрема, було протестовано:

- можливість запуску програми після декомпіляції та повторної збірки;
- коректність роботи механізмів автентифікації після модифікації клієнтського коду;
- залежність доступу до даних від перевірок сервера.

Результати показали, що навіть у разі успішної модифікації клієнтського застосунку, зловмисник не може отримати доступ до чатів або повідомлень без проходження автентифікації та дотримання правил Firestore. Це підтверджує ефективність підходу, в якому критичні перевірки безпеки виконуються на рівні сервера.

Тестування безпеки мережі. Тестування на рівні мережі включало аналіз переданого трафіку за допомогою проксі-інструментів. Було протестовано наступне:

- використання виключно безпечних TLS-з'єднань;
- відсутність відкритих даних у мережевих запитах;
- коректна обробка помилок мережевого з'єднання.

Аналіз трафіку показав, що всі мережеві взаємодії відбуваються через зашифровані канали, а передані дані додатково шифруються на рівні застосунку.

Стислий виклад результатів тестування. Застосована методологія тестування дозволила комплексно оцінити рівень безпеки розробленого мобільного застосунку. Проведені тести підтвердили:

- коректну реалізацію механізмів автентифікації;
- ефективний контроль доступу до даних;
- стійкість системи до несанкціонованого доступу;
- відповідність впроваджених заходів сучасним вимогам інформаційної безпеки.

Отримані результати свідчать про доцільність використання запропонованої методології тестування під час розробки безпечних Android-додатків та можуть бути використані як практичні рекомендації для подібних систем.

Тестування системи безпеки мобільного додатку є невід'ємною частиною процесу розробки, оскільки дозволяє оцінити ефективність реалізованих захисних механізмів та виявити потенційні вразливості. В рамках цієї магістерської роботи було застосовано комплексну методологію тестування, яка поєднує функціональні, інтеграційні та безпекові тести.

Функціональне тестування було спрямоване на перевірку коректності роботи основних сценаріїв використання додатку, зокрема процесів автентифікації користувачів, створення чатів, обміну повідомленнями та синхронізації даних у режимі реального часу. Особливу увагу було приділено сценаріям повторного входу користувача, втрати мережевого з'єднання та повторної ініціалізації сеансу, оскільки саме в таких умовах часто виникають помилки, які можуть призвести до витоку даних.

Тестування безпеки включало перевірку реалізованих правил доступу Firebase Firestore, які виступають у ролі серверного механізму контролю доступу. Було перевірено, що користувач не має можливості читати або змінювати дані в чатах, учасником яких він не є, навіть у разі модифікації клієнтського додатку. Для цього використовувалися сценарії заміни ідентифікаторів чату та спроб безпосереднього доступу до документів бази даних.

Окремо тестувалися механізми наскрізного шифрування. Для цього було проаналізовано вміст даних, що зберігаються у Firebase Firestore, щоб підтвердити відсутність відкритого тексту повідомлень. Отримані результати підтвердили, що всі повідомлення зберігаються в зашифрованому вигляді, а їх розшифрування можливе лише на стороні клієнта за наявності відповідних криптографічних ключів.

3.2. Перевірка застосунку на вразливості за OWASP Mobile Top 10

Для комплексної оцінки рівня захищеності розробленого мобільного застосунку використано рекомендації **OWASP Mobile Top 10**, які є одним із найбільш авторитетних і широко застосовуваних стандартів у сфері безпеки мобільних застосунків. Даний перелік охоплює найбільш критичні категорії вразливостей, що зустрічаються у сучасних Android- та iOS-застосунках, і використовується як основа для побудови безпечної архітектури та проведення безпекового тестування.

Перевірка здійснювалася шляхом аналізу архітектури застосунку, клієнтського коду, конфігурації хмарної інфраструктури Firebase, правил доступу Firestore, а також фактичної поведінки застосунку під час виконання типових та нештатних сценаріїв.

M1: Improper Platform Usage (Неправильне використання механізмів платформи)

Дана категорія охоплює помилки, пов’язані з некоректним використанням API операційної системи Android, механізмів дозволів, міжпроцесної взаємодії та системних компонентів.

У розробленому застосунку:

- не використовується небезпечна міжпроцесна взаємодія (implicit intents);
- доступ до системних ресурсів обмежений мінімально необхідними дозволами;
- криптографічні операції виконуються з використанням Android Keystore;
- не застосовується динамічне завантаження виконуваного коду.

Таким чином, ризики, пов’язані з неправильним використанням механізмів платформи, зведені до мінімуму.

M2: Insecure Data Storage (Небезпечне зберігання даних)

Вразливості цієї категорії виникають у разі зберігання конфіденційної інформації у відкритому вигляді у файловій системі пристрою або локальних сховищах.

У межах розробленого застосунку:

- текстові повідомлення не зберігаються у відкритому вигляді ані локально, ані у хмарному сховищі;
- у Firestore зберігаються виключно зашифровані повідомлення;
- криптографічні ключі не зберігаються у файловій системі застосунку, а ізольовані у Android Keystore;
- токени автентифікації обробляються виключно через Firebase Authentication.

Отже, застосунок відповідає вимогам OWASP щодо безпечного зберігання даних.

M3: Insecure Communication (Небезпечна передача даних)

Ця категорія охоплює вразливості, пов’язані з передачею даних через незахищені або неправильно налаштовані мережеві канали.

У розробленому прототипі:

- усі мережеві з’єднання здійснюються через TLS;
- використовується офіційний Firebase SDK, який забезпечує перевірку сертифікатів;
- дані додатково шифруються на прикладному рівні до передачі мережею.

Навіть у разі компрометації транспортного рівня зловмисник не зможе отримати доступ до вмісту повідомлень, що свідчить про відповідність вимогам OWASP щодо захищеної комунікації.

M4: Insecure Authentication (Небезпечна автентифікація)

Уразливості цієї категорії виникають через слабкі або некоректно реалізовані механізми автентифікації.

У розробленому застосунку реалізовано:

- автентифікацію за електронною поштою та паролем;
- обов’язкову верифікацію електронної пошти;
- використання захищених токенів доступу Firebase;
- можливість застосування біометричної автентифікації як додаткового рівня захисту.

Такий підхід відповідає принципам багатофакторної автентифікації та значно зменшує ризик компрометації облікових записів.

M5: Insufficient Cryptography (Недостатня криптографія)

Недостатня або неправильно реалізована криптографія є однією з найнебезпечніших категорій вразливостей.

У розробленому застосунку:

- використовується ECDH для узгодження ключів;

- застосовується сучасне симетричне шифрування для повідомлень;
- ключі не передаються мережею і не зберігаються у відкритому вигляді;
- додатково використовується контекстна прив’язка ключів до ідентифікатора чату.

Реалізована криптографічна схема відповідає сучасним вимогам і не містить відомих слабких алгоритмів.

M6: Insecure Authorization (Небезпечна авторизація)

Ця категорія пов’язана з помилками контролю доступу до ресурсів.

У застосунку:

- всі перевірки доступу виконуються на серверному рівні за допомогою правил Firestore;
- клієнтський код не має вирішального впливу на авторизацію;
- доступ до чатів і повідомлень дозволений виключно учасникам відповідного діалогу.

Таким чином, ризики несанкціонованого доступу до даних мінімізовані.

M7: Client Code Tampering (Модифікація клієнтського коду)

Для протидії модифікації клієнтського коду:

- застосовано обфускацію (R8/ProGuard);
- мінімізовано критичну логіку на стороні клієнта;
- серверна частина виконує незалежну перевірку прав доступу.

Навіть у разі модифікації APK доступ до даних без автентифікації залишається неможливим.

M8: Reverse Engineering (Реверс-інжиніринг)

Обфускація, оптимізація коду та використання стандартних SDK суттєво ускладнюють аналіз застосунку. Крім того, відсутність секретів у клієнтському коді зменшує практичну цінність результатів реверс-інжинірингу.

M9: Extraneous Functionality (Зайва функціональність)

У застосунку відсутні приховані тестові ендпойнти, дебаг-інтерфейси або службова функціональність, яка могла б бути використана зловмисником.

Узагальнення результатів перевірки за OWASP Mobile Top 10

Проведений аналіз показав, що розроблений мобільний застосунок:

- відповідає більшості вимог OWASP Mobile Top 10;
- ефективно протидіє основним класам мобільних загроз;
- демонструє високий рівень захищеності даних і облікових записів користувачів.

У сукупності це дозволяє зробити висновок, що застосунок відповідає вимогам **OWASP MASVS рівня L1**, а окремі механізми — рівня **L2**, що є високим показником для мобільного застосунку навчально-практичного призначення.

Для оцінки рівня захищеності розробленого мобільного застосунку використано рекомендації OWASP Mobile Top 10, які є одним з найбільш авторитетних міжнародних стандартів у сфері безпеки мобільних застосунків. Аналіз проводився шляхом зіставлення реалізованих механізмів захисту з основними класами загроз, визначеними у даному стандарті.

Загрози, пов’язані з некоректним використанням можливостей платформи (M1: Improper Platform Usage), мінімізовані завдяки використанню стандартних API Android та відмові від небезпечних практик, таких як динамічне завантаження коду або виконання коду з зовнішніх джерел. Захист локальних даних (M2: Insecure Data Storage) забезпечено шляхом відмови від зберігання чутливої інформації у відкритому вигляді та використання Android Keystore для керування криптографічними ключами.

Загрози, пов’язані з передачею даних (M3: Insecure Communication), усуваються шляхом використання захищених мережевих з’єднань та клієнтського шифрування повідомлень. Навіть у разі компрометації мережевого каналу зловмисник не може отримати доступ до вмісту повідомлень. Недостатня криптографія (M5: Insufficient Cryptography)

усунена завдяки використанню сучасних криптографічних алгоритмів та перевірених схем обміну ключами.

Таким чином, розроблений прототип відповідає основним вимогам OWASP Mobile Application Security Verification Standard на базовому та розширеному рівнях і демонструє високий рівень захищеності від найбільш поширених класів атак.

3.3. Порівняння рівня захисту прототипу з існуючими мобільними додатками

Для оцінки ефективності та практичної значущості реалізованого прототипу доцільно провести порівняльний аналіз його рівня безпеки з поширеними мобільними застосунками для обміну повідомленнями. Таке порівняння дозволяє визначити сильні та слабкі сторони запропонованих рішень, а також оцінити відповідність сучасним вимогам інформаційної безпеки.

Порівняння здійснювалося за низкою ключових критеріїв, які є визначальними для безпеки мобільних чатів: автентифікація користувачів, захист даних, контроль доступу, захист клієнтського коду та архітектурні особливості реалізації.

Критерії порівняння

Для аналізу було обрано такі критерії:

- методи автентифікації та керування сесіями;
- наявність і реалізація багатофакторної автентифікації;
- захист мережевих з’єднань;
- спосіб зберігання та обробки повідомлень;
- використання наскрізного шифрування;
- контроль доступу до даних;
- стійкість до реверс-інжинірингу та модифікації клієнтського коду.

Порівняння з типовими комерційними месенджерами

Більшість популярних комерційних месенджерів використовують централізовану архітектуру, у якій серверна частина відіграє ключову роль у зберіганні та обробці повідомлень. У таких системах, навіть за наявності захищених TLS-з'єднань, сервер часто має доступ до відкритого тексту повідомлень або може здійснювати їхню обробку для додаткових сервісів, таких як пошук, аналітика або резервне копіювання.

На відміну від цього підходу, у розробленому прототипі реалізовано **наскрізне шифрування**, за якого серверна інфраструктура не має доступу до вмісту повідомлень. Повідомлення шифруються на стороні клієнта та зберігаються у хмарному сховищі виключно у зашифрованому вигляді. Це забезпечує вищий рівень конфіденційності та зменшує ризики, пов'язані з компрометацією серверної частини.

Автентифікація та контроль доступу

У більшості існуючих мобільних застосунків автентифікація базується на поєднанні логіну та пароля або номеру телефону. Додаткові фактори автентифікації часто є опціональними та можуть бути вимкнені користувачем.

У розробленому прототипі автентифікація реалізована з використанням Firebase Authentication і доповнена обов'язковою верифікацією електронної пошти. Контроль доступу до даних здійснюється не на клієнтському рівні, а за допомогою строгих правил безпеки Firestore, що перевіряються серверною частиною при кожному запиті. Це забезпечує більш надійний захист у порівнянні з підходами, які покладаються на логіку клієнтського застосунку.

Захист даних та криптографічні механізми

У багатьох мобільних чатах шифрування повідомлень реалізується лише на транспортному рівні. Хоча такий підхід захищає дані від перехоплення під час передачі, він не гарантує їхню конфіденційність у разі доступу до серверної інфраструктури.

Розроблений прототип використовує поєднання асиметричних і симетричних криптографічних алгоритмів, що дозволяє реалізувати повноцінне end-to-end шифрування. Криптографічні ключі зберігаються у Android Keystore, що значно підвищує стійкість системи до компрометації ключового матеріалу.

Архітектурні відмінності та їх вплив на безпеку

Важливою відмінністю розробленого застосунку є його архітектура, орієнтована на принципи «security by design» та «defense in depth». Безпека

інтегрована у всі рівні системи: від клієнтського інтерфейсу до серверних правил доступу.

На відміну від багатьох існуючих застосунків, де безпека додається як окремий модуль або опціональна функція, у даному прототипі вона є невід'ємною частиною архітектури. Це зменшує кількість потенційних вразливостей та підвищує стійкість системи до комбінованих атак.

Порівняльна оцінка рівня захисту

За сукупністю розглянутих критеріїв розроблений прототип демонструє рівень захищеності, який не поступається сучасним комерційним рішенням, а за окремими аспектами — зокрема конфіденційністю повідомлень і прозорістю архітектури — має переваги. Водночас слід зазначити, що комерційні месенджери можуть мати додаткові механізми захисту, пов'язані з масштабуванням, резервним копіюванням та централізованим моніторингом безпеки.

Висновки за результатами порівняння

Проведене порівняння показало, що розроблений мобільний застосунок:

- забезпечує високий рівень конфіденційності переданих повідомлень;
- реалізує строгий контроль доступу до даних;
- використовує сучасні криптографічні та архітектурні підходи;
- відповідає рекомендаціям OWASP Mobile Top 10 та MASVS.

Отримані результати підтверджують доцільність використання запропонованої архітектури та реалізованих механізмів безпеки як основи для створення захищених мобільних застосунків обміну повідомленнями.

Для оцінки практичної цінності розробленого прототипу виконано порівняльний аналіз із поширеними мобільними застосунками для обміну повідомленнями. Основними критеріями порівняння були механізми автентифікації, підходи до зберігання даних, рівень шифрування повідомлень та контроль доступу.

Більшість комерційних месенджерів використовують шифрування під час передачі даних, однак не завжди реалізують повноцінне наскрізне шифрування або дозволяють серверній частині доступ до відкритого тексту

повідомлень. У розробленому прототипі серверна інфраструктура не має доступу до вмісту повідомлень, що значно підвищує рівень конфіденційності.

Крім того, використання строгих правил безпеки Firestore забезпечує точковий контроль доступу до даних на рівні окремих документів і підколекцій. Це унеможливлює несанкціонований доступ навіть у разі компрометації клієнтського застосунку, що не завжди характерно для типових реалізацій мобільних чатів.

3.4. Аналіз продуктивності та впливу заходів безпеки на швидкодію

Запровадження багаторівневих механізмів безпеки у мобільних застосунках неминуче впливає на їхню продуктивність та споживання ресурсів. Зокрема, криптографічні операції, додаткові перевірки доступу та мережеві механізми захисту можуть створювати обчислювальні та часові накладні витрати. Тому важливим етапом дослідження є аналіз впливу реалізованих заходів безпеки на швидкодію розробленого Android-застосунку та оцінка їхнього впливу на користувацький досвід.

Методологія аналізу продуктивності

Аналіз продуктивності проводився у реальних умовах експлуатації застосунку з використанням фізичних Android-пристроїв середнього класу та стандартних інструментів профілювання Android Studio. Оцінка здійснювалася за такими показниками:

- час автентифікації користувача;
- затримка під час відкриття списку чатів;
- час надсилання та отримання повідомлень;
- затримка, пов’язана з шифруванням і розшифруванням даних;
- стабільність роботи інтерфейсу під час активного обміну повідомленнями.

Для забезпечення об’єктивності результати порівнювалися з базовими сценаріями, у яких аналогічна функціональність реалізована без клієнтського наскрізного шифрування.

Вплив криптографічних операцій на швидкодію

Найбільш ресурсомістким елементом системи безпеки є криптографічні операції, зокрема генерація ключів, узгодження симетричного ключа та

безпосереднє шифрування і розшифрування повідомлень. У розробленому застосунку асиметричні операції виконуються лише у визначені моменти, зокрема під час першої ініціалізації чату або повторної генерації ключів.

Завдяки використанню Android Keystore та оптимізованих криптографічних бібліотек, операції узгодження ключів виконуються швидко та не блокують головний потік інтерфейсу. Симетричне шифрування повідомлень здійснюється локально у фонових потоках і не створює помітних затримок навіть при активному обміні повідомленнями.

Проведені вимірювання показали, що затримка, пов’язана з шифруванням одного текстового повідомлення, є незначною у порівнянні з часом мережевої взаємодії та не впливає на сприйняття швидкодії користувачем.

Вплив мережевих механізмів безпеки

Використання захищених TLS-з’єднань є стандартом для сучасних мобільних застосунків і не створює суттєвих накладних витрат у порівнянні з незахищеними з’єднаннями. Firebase SDK оптимізує встановлення та повторне використання мережевих сесій, що дозволяє мінімізувати затримки при обміні даними.

Аналіз показав, що основний внесок у затримку доставки повідомлень вносить саме мережеве середовище, а не криптографічні механізми або правила доступу. При стабільному інтернет-з’єднанні повідомлення відображаються у реальному часі без відчутних затримок.

Вплив серверних перевірок доступу

Реалізація строгих правил безпеки Firestore передбачає перевірку прав доступу до кожного запиту на читання або запис даних. Ці перевірки виконуються на серверному рівні та не залежать від обчислювальних ресурсів клієнтського пристрою.

Практичні спостереження показали, що серверні перевірки доступу не мають помітного впливу на час відповіді при роботі з чатами, навіть за умови активної синхронізації даних. Це підтверджує доцільність перенесення критичної логіки контролю доступу з клієнтської частини на серверну.

Вплив безпеки на користувацький досвід

Одним із ключових завдань при розробці захищених застосунків є збереження позитивного користувацького досвіду. У розробленому прототипі всі операції, пов’язані з безпекою, виконуються прозоро для користувача та не потребують додаткових дій у повсякденному використанні.

Процес автентифікації та підтвердження електронної пошти відбувається лише під час першого входу або реєстрації. Подальша робота з чатами не ускладнюється додатковими запитами або підтвердженнями. Біометрична автентифікація, за необхідності, використовується лише як додатковий захисний рівень і не впливає на основні сценарії взаємодії.

Узагальнення результатів аналізу

Результати аналізу продуктивності показали, що реалізовані заходи безпеки:

- не створюють суттєвих затримок у роботі застосунку;
- не погіршують користувацький досвід;
- ефективно масштабуються разом зі збільшенням кількості користувачів і повідомлень;
- забезпечують високий рівень захисту без критичного впливу на швидкодію.

Таким чином, багаторівнева система безпеки, реалізована у розробленому мобільному застосунку, демонструє оптимальний баланс між рівнем захищеності та продуктивністю, що є важливою вимогою для сучасних мобільних інформаційних систем.

Запровадження криптографічних механізмів неминуче впливає на продуктивність мобільного застосунку, тому в межах роботи проведено аналіз цього впливу. Основні криптографічні операції, зокрема генерація ключів та узгодження симетричних ключів, виконуються нечасто і не впливають на повсякденну роботу користувача.

Шифрування та розшифрування повідомлень здійснюється локально на пристрої користувача і виконується асинхронно, що дозволяє уникнути блокування головного потоку інтерфейсу. Під час тестування не було зафіксовано помітних затримок у роботі застосунку при обміні короткими текстовими повідомленнями.

Отримані результати свідчать про те, що реалізовані заходи безпеки забезпечують високий рівень захисту без суттєвого погіршення користувацького досвіду, що є важливим чинником для практичного використання мобільних застосунків.

3.5. Рекомендації щодо архітектури застосунків та захисту даних

На основі проведеного теоретичного аналізу, практичної реалізації прототипу мобільного застосунку та результатів тестування системи безпеки можна сформулювати низку рекомендацій щодо побудови захищених Android-застосунків. Запропоновані рекомендації орієнтовані на розробників мобільного програмного забезпечення, які працюють із конфіденційними даними користувачів, а також можуть бути використані під час проєктування корпоративних або публічних мобільних інформаційних систем.

Архітектурні рекомендації

Першочерговим завданням при створенні безпечного мобільного застосунку є коректне архітектурне проєктування. Рекомендується застосовувати принцип **security by design**, відповідно до якого механізми безпеки мають інтегруватися у систему з самого початку, а не додаватися на пізніх етапах розробки.

Доцільно використовувати архітектурні підходи, які забезпечують чітке розмежування відповідальності між компонентами, зокрема MVVM або MVI. Це дозволяє ізолювати бізнес-логіку, пов’язану з безпекою, від користувацького інтерфейсу та зменшити ризик помилок реалізації. Критично важливо, щоб перевірки доступу та автентифікації не залежали від стану UI-компонентів.

Також рекомендовано мінімізувати обсяг критичної логіки на стороні клієнта та переносити контроль доступу на серверний рівень. Клієнтський застосунок не повинен мати вирішального впливу на прийняття рішень щодо доступу до даних.

Рекомендації щодо захисту даних

Захист даних користувачів повинен здійснюватися на всіх етапах їхнього життєвого циклу: під час створення, передачі, обробки та зберігання. Для цього доцільно застосовувати багаторівневий підхід, який поєднує мережеве шифрування та прикладну криптографію.

Рекомендується використовувати **наскрізне шифрування** для обміну конфіденційними даними, особливо у застосунках для спілкування, фінансових операцій або роботи з персональною інформацією. Криптографічні ключі мають зберігатися у захищеному середовищі пристрою з використанням Android Keystore, а не у файловій системі застосунку.

Необхідно уникати зберігання відкритих даних у локальних сховищах, кеші або логах. Навіть службова інформація повинна аналізуватися з точки зору можливого витоку чутливих відомостей.

Рекомендації щодо автентифікації та авторизації

Для підвищення рівня безпеки облікових записів рекомендується використовувати багатофакторну автентифікацію або її елементи, зокрема підтвердження електронної пошти, одноразові коди або біометричні методи. Навіть базове підтвердження email-адреси суттєво зменшує ризик використання фіктивних або автоматично створених акаунтів.

Авторизація доступу до даних повинна здійснюватися незалежно від клієнтського застосунку. Серверна частина має перевіряти кожен запит на відповідність політикам доступу. У разі використання хмарних сервісів доцільно застосовувати декларативні правила безпеки, які чітко визначають, хто і за яких умов може читати або змінювати дані.

Рекомендації щодо захисту клієнтського коду

Для ускладнення реверс-інжинірингу та модифікації клієнтського коду доцільно використовувати обфускацію, оптимізацію байткоду та видалення невикористовуваних компонентів. Водночас не слід покладатися виключно на обфускацію як на основний механізм захисту, оскільки вона лише ускладнює аналіз, але не усуває саму можливість атаки.

Рекомендується уникати зберігання секретів, ключів або критичних параметрів конфігурації у клієнтському коді. Усі чутливі значення повинні або динамічно отримуватися з захищених джерел, або зберігатися у захищеному середовищі платформи.

Рекомендації щодо тестування та підтримки безпеки

Забезпечення безпеки мобільного застосунку не є одноразовою дією, а безперервним процесом. Рекомендується регулярно проводити тестування на відповідність рекомендаціям OWASP Mobile Top 10 та Mobile Application Security Verification Standard.

Доцільно поєднувати автоматизоване тестування з ручним аналізом архітектури та сценаріїв використання. Особливу увагу слід приділяти оновленням бібліотек і залежностей, оскільки застарілі компоненти можуть містити відомі вразливості.

Узагальнення рекомендацій

Запропоновані рекомендації дозволяють сформувати цілісний підхід до побудови захищених Android-застосунків, у якому безпека розглядається як інтегрована властивість системи, а не як додатковий функціональний модуль. Використання багаторівневої архітектури, сучасних криптографічних механізмів та перевірених хмарних сервісів забезпечує високий рівень

захисту даних без істотного погіршення продуктивності або зручності користування.

На основі проведеного дослідження та практичної реалізації можна сформулювати низку рекомендацій щодо розробки безпечних Android-застосунків. Насамперед доцільно впроваджувати багаторівневий підхід до безпеки, який поєднує захист коду, даних, мережевих з’єднань та механізмів автентифікації.

Рекомендується використовувати системні механізми захисту платформи Android, зокрема Android Keystore, та уникати зберігання чутливої інформації у відкритому вигляді. Для застосунків, що працюють з конфіденційними даними, доцільно реалізовувати наскрізне шифрування, яке унеможливлює доступ до даних з боку серверної інфраструктури.

Регулярна перевірка відповідності застосунку рекомендаціям OWASP Mobile Top 10 та використання автоматизованих і ручних методів тестування дозволяє своєчасно виявляти та усувати потенційні вразливості. Запропоновані рекомендації можуть бути використані як основа для створення безпечних мобільних застосунків у різних предметних галузях.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було досліджено проблему забезпечення інформаційної безпеки мобільних застосунків на платформі Android та обґрунтовано доцільність використання багаторівневого підходу до захисту даних і функціональності таких систем. Проведений аналіз сучасних загроз інформаційній безпеці мобільних застосунків показав, що реалізація окремих захисних механізмів не забезпечує належного рівня захисту і потребує комплексного поєднання клієнтських та серверних засобів безпеки.

У межах роботи було розроблено та реалізовано прототип мобільного Android-застосунку для безпечного обміну текстовими повідомленнями, який створено спеціально для даного дослідження. Прототип реалізовано з використанням мови програмування Kotlin, сучасного інтерфейсу Jetpack Compose та хмарної платформи Firebase. Реалізований застосунок слугує практичною основою для апробації та аналізу сучасних механізмів захисту інформації.

Для підвищення рівня інформаційної безпеки у розробленому застосунку реалізовано багаторівневу систему захисту, яка включає захист клієнтського коду, даних, мережевих з’єднань та механізмів автентифікації користувачів. Захист клієнтського коду забезпечено шляхом використання обфускації та мінімізації критичної логіки на стороні мобільного застосунку, що ускладнює реверс-інжиніринг та модифікацію програмного коду.

Захист даних реалізовано із застосуванням сучасних криптографічних методів. Для зберігання та використання криптографічних ключів використано Android Keystore, що забезпечує ізоляцію приватних ключів у захищеному середовищі пристрою. Реалізовано наскрізне шифрування повідомлень на основі алгоритму обміну ключами ECDH та симетричного шифрування AES-GCM, що унеможливлює доступ до вмісту повідомлень з боку серверної інфраструктури.

Механізми автентифікації та контролю доступу реалізовано з використанням Firebase Authentication та правил безпеки Firebase Firestore. Це забезпечує надійний серверний контроль доступу до даних та унеможливлює несанкціонований доступ до чатів і повідомлень навіть у разі компрометації клієнтського застосунку. Захист мережевих з’єднань реалізовано шляхом використання захищених каналів зв’язку та поєднання транспортного захисту з клієнтським шифруванням даних.

У ході тестування підтверджено ефективність реалізованих заходів безпеки та їх відповідність рекомендаціям OWASP Mobile Top 10 і Mobile Application Security Verification Standard. Проведений аналіз показав, що застосування багаторівневого підходу дозволяє значно знизити ризики

компрометації даних користувачів без суттєвого впливу на продуктивність і зручність використання застосунку.

Практична цінність отриманих результатів полягає у можливості використання розробленого прототипу та запропонованих підходів як основи для створення безпечних мобільних Android-застосунків, що працюють з конфіденційними даними, а також у навчальних і дослідницьких цілях.

ПЕРЕЛІК ПОСИЛАНЬ

1. Android Developers. Security overview. URL: https://developer.android.com/security (дата звернення: 05.11.2025)
2. Android Developers. Privacy and security overview. URL: https://developer.android.com/privacy-and-security/about (дата звернення: 06.11.2025).
3. Android Developers. Privacy and security risks. URL: https://developer.android.com/privacy-and-security/risks (дата звернення: 07.11.2025).
4. Android Developers. Insecure API usage. URL: https://developer.android.com/privacy-and-security/risks/insecure-api-usage (дата звернення: 08.11.2025).
5. Android Developers. Implicit intent hijacking. URL: https://developer.android.com/privacy-and-security/risks/implicit-intent-hijacking (дата звернення: 09.11.2025).
6. Android Developers. Dynamic code loading. URL: https://developer.android.com/privacy-and-security/risks/dynamic-code-loading (дата звернення: 10.11.2025).
7. Android Developers. Create package context. URL: https://developer.android.com/privacy-and-security/risks/create-package-context (дата звернення: 11.11.2025).
8. Android Developers. Android Keystore system. URL: https://developer.android.com/privacy-and-security/keystore (дата звернення: 12.11.2025).
9. Android Developers. Cryptography overview. URL: https://developer.android.com/privacy-and-security/cryptography (дата звернення: 13.11.2025).
10. Android Developers. Network security configuration. URL: https://developer.android.com/privacy-and-security/security-config (дата звернення: 15.11.2025).
11. Android Developers. App integrity. URL: https://developer.android.com/google/play/integrity (дата звернення: 16.11.2025).
12. Android Developers. Biometric authentication. URL: https://developer.android.com/training/sign-in/biometric-auth (дата звернення: 17.11.2025).
13. Android Developers. Data and file storage overview. URL: https://developer.android.com/training/data-storage (дата звернення: 18.11.2025).
14. Android Developers. Permissions overview. URL: https://developer.android.com/guide/topics/permissions/overview (дата звернення: 19.11.2025).
15. Android Developers. Background work limits. URL:

https://developer.android.com/about/versions/oreo/background
(дата звернення: 20.11.2025).
16. OWASP Foundation. OWASP Mobile Top 10. URL:
https://owasp.org/www-project-mobile-top-10/
(дата звернення: 22.11.2025).
17. OWASP Foundation. Mobile Security Testing Guide. URL:
https://owasp.org/www-project-mobile-security-testing-guide/
(дата звернення: 24.11.2025).
18. Google Developers. Firebase Authentication documentation. URL:
https://firebase.google.com/docs/auth
(дата звернення: 26.11.2025).
19. Google Developers. Firebase email verification. URL:
https://firebase.google.com/docs/auth/android/manage-users
(дата звернення: 27.11.2025).
20. Google Developers. Firebase Firestore documentation. URL:
https://firebase.google.com/docs/firestore
(дата звернення: 28.11.2025).
21. Google Developers. Firestore security rules. URL:
https://firebase.google.com/docs/firestore/security/get-started
(дата звернення: 29.11.2025).
22. Google Developers. Firebase security rules overview. URL:
https://firebase.google.com/docs/rules
(дата звернення: 01.12.2025).
23. Google Developers. Firebase App Check. URL:
https://firebase.google.com/docs/app-check
(дата звернення: 05.12.2025).
24. Google Developers. Firebase best practices for security. URL:
https://firebase.google.com/docs/database/security
(дата звернення: 07.12.2025).
25. NIST. Digital Identity Guidelines (SP 800-63). URL:
https://pages.nist.gov/800-63-3/
(дата звернення: 09.12.2025).
26. NIST. Cryptographic standards and guidelines. URL:
https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines
(дата звернення: 11.12.2025).
27. Cloudflare. What is TLS? URL:
https://www.cloudflare.com/learning/ssl/what-is-tls/
(дата звернення: 13.12.2025).
28. Cloudflare. End-to-end encryption explained. URL:
https://www.cloudflare.com/learning/privacy/what-is-end-to-end-encryption/
(дата звернення: 15.12.2025).
29. JetBrains. Kotlin language documentation. URL:
https://kotlinlang.org/docs/home.html
(дата звернення: 19.12.2025).

30. OWASP Foundation. Mobile Application Security Verification Standard – Storage Requirements. URL: https://mas.owasp.org/MASVS/05-MASVS-STORAGE/ (дата звернення: 19.12.2025).
31. Android Developers. Security overview. URL: https://developer.android.com/security (дата звернення: 19.12.2025).
32. OWASP Foundation. Mobile Application Security Verification Standard (MASVS) – PDF version. URL: https://github.com/OWASP/owasp-masvs/releases/latest/download/OWASP_MASVS.pdf (дата звернення: 19.12.2025).
33. Android Developers. Privacy and security risks overview. URL: https://developer.android.com/privacy-and-security/risks (дата звернення: 20.12.2025).
34. Android Developers. Hardcoded cryptographic secrets. URL: https://developer.android.com/privacy-and-security/risks/hardcoded-cryptographic-secrets (дата звернення: 21.12.2025).
35. Android Developers. Android security cheat sheet. URL: https://developer.android.com/privacy-and-security/images/cheat-sheet-light.pdf (дата звернення: 22.12.2025).
36. Android Developers. Security tips. URL: https://developer.android.com/privacy-and-security/security-tips (дата звернення: 22.12.2025).

ДОДАТКИ

Додаток А

Програмна реалізація мобільного застосунку, розробленого в межах магістерської кваліфікаційної роботи, доступна у відкритому репозиторії GitHub за посиланням:

https://github.com/altsanityqq/Android-Messaging-Application