**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО**
**«ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»**
**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ**
**КАФЕДРА КІБЕРБЕЗПЕКИ**

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
"_____"_____ 20__ р.

# КВАЛІФІКАЦІЙНА РОБОТА

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТР"**

**Тема:** Програмний застосунок менеджера паролів на основі біометричної автентифікації користувачів

**Виконавець:** Богдан ХУКАЛЕНКО

**Керівник:** к.т.н. Олена ВИСОЦЬКА

**Нормоконтролер:** к.т.н., доцент Андрій ПЕТРЕНКО

**Київ 2024**

**ДЕРЖАВНЕ НЕКОМЕРЦІЙНЕ ПІДПРИЄМСТВО**
**«ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**«КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»**

Факультет комп'ютерних наук та технологій
Кафедра кібербезпеки
Освітній ступінь магістр
Спеціальність 125 «Кібербезпека та захист інформації»
Освітньо-професійна програма «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки

_____Анна ІЛЬЄНКО
«30»___08_____2024 р.

**ЗАВДАННЯ**

**на виконання кваліфікаційної роботи**

**Хукаленко Богдана Андрійовича**

1. Тема кваліфікаційної роботи: Програмний застосунок менеджера паролів на основі біометричної автентифікації користувачів
затверджена наказом ректора від 30.08.2024 р. №1695/ст.
2. Термін виконання роботи: з 30.08.2024 по 15.12.2024
3. Вихідні дані до роботи: проаналізувати існуючі системи менеджерів паролів; обрати метод автентифікації; дослідити технології біометричної автентифікації; обрати метод біометричної автентифікації; розробити програмний застосунок менеджера паролів; протестувати функціональні можливості розробленого застосунку.
4. Зміст пояснювальної записки: аналіз сучасних технологій з управління паролів; аналіз методів біометричної автентифікації; технічна реалізація та алгоритми роботи застосунку; програмний застосунок менеджера паролів на основі біометричної автентифікації користувачів.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: презентація.

6. Календарний план-графік

| № з/п | Завдання | Термін виконання | Підпис керівника |
|---|---|---|---|
| 1. | Дослідити сучасні методи автентифікації. | 30.08.2024 – 05.09.2024 | *Виконано* |
| 2. | Провести аналіз методів біометричної автентифікації користувачів | 05.09.2024 – 10.09.2024 | *Виконано* |
| 3. | Обґрунтувати вибір автентифікації за відбитком пальця | 11.09.2024 – 14.09.2024 | *Виконано* |
| 4. | Провести аналіз сучасних рішень з управління паролів. | 15.09.2024 – 19.09.2024 | *Виконано* |
| 5. | Провести аналіз методу атрибутивного шифрування. | 19.09.2024 – 22.09.2024 | *Виконано* |
| 6. | Розробити методику та алгоритм реалізації програмного застосунку менеджера паролів на біометричної автентифікації | 23.09.2024 – 27.09.2024 | *Виконано* |
| 7. | Створити програмне забезпечення запропонованої системи | 28.09.2024 – 14.10.2024 | *Виконано* |
| 8. | Протестувати роботу створеного застосунку, порівняти з існуючими аналогами | 15.10.2024 – 25.10.2024 | *Виконано* |

7. Дата видачі завдання: «30» __08__ 2024 р.

Керівник кваліфікаційної роботи: _____ Олена ВИСОЦЬКА
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання: _____ Богдан ХУКАЛЕНКО
(підпис здобувача вищої освіти) (П.І.Б.)

# РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний застосунок менеджера паролів на основі біометричної автентифікації користувачів»: робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків та має 117 с., 23 рис., 4 табл., 22 літературних джерела.

Об’єкт дослідження: процес управління паролями та процес автентифікації в сучасних інформаційних системах.

Предмет дослідження: технології менеджменту паролів, методи біометричної автентифікації, технології атрибутивного шифрування

Мета кваліфікаційної роботи: розробити програмний застосунок менеджер паролів, в якому завдяки біометричній автентифікації та атрибутивному шифруванню з перевіркою геолокації досягається новий рівень безпеки даних користувачів.

Методи дослідження: базуються на основі системного аналізу предметної області досліджень та об’єктно-орієнтованого програмування задля програмної реалізації розробленого застосунку.

Практична цінність: розроблено програмний застосунок на основі біометричної автентифікації користувачів призначений для менеджменту паролів. Він може бути використаний в мобільних пристроях на підприємствах для збереження конфіденційної інформації про паролі в безпечному середовищі та розмежування доступу до даних серед користувачів. Окрім цього, застосунок можна використовувати в домашньому середовищі як додатковий засіб забезпечення батьківського контролю. Використання застосунку дозволяє мати зручний доступ до інформації про паролі, котра буде надійно захищена завдяки поєднанню атрибутивного шифрування, перевірки геолокації та біометричних даних.

Наукова новизна: запропоновано авторський програмний застосунок менеджера паролів на основі біометричної автентифікації з використанням

атрибутивного шифрування та геолокації, що дозволяє посилити безпеку даних та комфорт користувачів.

Результати кваліфікаційної роботи рекомендується використовувати для забезпечення додаткової безпеки та зручності доступу на підприємствах та при особистому використанні.

МЕНЕДЖЕР ПАРОЛІВ, БІОМЕТРИЧНА АВТЕНТИФІКАЦІЯ, ВІДБИТОК ПАЛЬЦЯ, ШИФРУВАННЯ, ГЕОЛОКАЦІЯ.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

| | | |
|---|---|---|
| ABE | – | Attribute-Based Encryption |
| CIA | – | Confidentiality, integrity, availability |
| ID | – | Identifier |
| IDS | – | Intrusion Detection System |
| IPS | – | Intrusion Prevention System |
| JPBC | – | Java Pairing-Based Cryptography |
| OTP | – | One-Time Password |
| UI | – | User Interface |
| VPN | – | Virtual Private Network |

# ВСТУП

**Актуальність теми.** У сучасному світі використання паролів стало невід'ємною частиною життя: вони захищають банківські акаунти, хмарні сервіси, профілі в соціальних мережах, а також файли, які містять особисту або цінну інформацію. До того ж, доступ до більшості інтернет-ресурсів здійснюється через облікові записи, які також вимагають введення пароля. Тому питання забезпечення надійності паролів, їх збереження та захисту від несанкціонованого доступу є однією з ключових проблем у сфері інформаційної безпеки.

Багато людей побоюються забути свої паролі та втратити доступ до важливих даних чи ресурсів. Через це вони часто обирають надто прості комбінації або використовують один і той самий пароль для кількох платформ, що значно підвищує ризик зламу. Менеджери паролів пропонують вирішення цієї проблеми.

Сучасні менеджери паролів не лише дозволяють генерувати складні та безпечні паролі, які відповідають найвищим стандартам захисту, але й надають можливість зберігати їх у надійному місці. Такі сервіси забезпечують конфіденційність, цілісність, доступність, невідмовність та контроль доступу до паролів, роблячи управління ними зручним і безпечним.

Біометрична автентифікація – сучасний метод захисту даних, що набуває дедалі більшої популярності та розповсюдженості. Ідентифікація за біометричними характеристиками забезпечує швидкий, зручний та надійний метод авторизації користувачів, та в комбінації з передовими методами шифрування забезпечує надзвичайно високий рівень безпеки інформації. Саме тому розробка програмного застосунку менеджеру паролів, який базується на біометричній автентифікації та використовує метод атрибутивного шифрування має вискі перспективи.

**Мета і завдання виконання кваліфікаційної роботи.** Метою роботи є розробити програмний застосунок менеджер паролів, в якому завдяки біометричній автентифікації та атрибутивному шифруванню з перевіркою геолокації досягається новий рівень безпеки даних користувачів.

Для досягнення поставленої мети необхідно розв'язати наступні задачі:

1) Проаналізувати сучасні методи автентифікації та біометричної ідентифікації та на основі результату проведеного аналізу обрати оптимальні технології для вирішення задачі з розробки менеджера паролів.

2) Розробити програмний застосунок менеджер паролів на основі біометричної автентифікації з використанням атрибутивного шифрування та геолокації, що забезпечить підвищений рівень захисту даних користувачів.

3) Провести тестування розробленого програмного застосунку, що дасть змогу дослідити доцільність використання розробленого застосунку для вирішення поставленої задачі.

**Об'єкт дослідження** – процес управління паролями та процес автентифікації в сучасних інформаційних системах.

**Предмет дослідження** – технології менеджменту паролів, методи біометричної автентифікації, технології атрибутивного шифрування.

**Методи дослідження**. Проведені дослідження базуються на основі системного аналізу предметної області досліджень та об’єктно-орієнтованого програмування задля програмної реалізації розробленого застосунку.

**Наукова новизна отриманих результатів** полягає в тому, що запропоновано авторський програмний застосунок менеджера паролів на основі біометричної автентифікації з використанням атрибутивного шифрування та геолокації, що дозволяє покращити безпеку даних та комфорт користувачів..

**Практичне значення отриманих результатів.** Розроблено програмний застосунок на основі біометричної автентифікації користувачів призначений для менеджменту паролів. Він може бути використаний в мобільних пристроях на підприємствах для збереження конфіденційної інформації про паролі в безпечному середовищі та розмежування доступу до даних серед користувачів.

Окрім цього, застосунок можна використовувати в домашньому середовищі як додатковий засіб забезпечення батьківського контролю. Використання застосунку дозволяє мати зручний доступ до інформації про паролі, котра буде надійно захищена завдяки поєднанню атрибутивного шифрування, перевірки геолокації та біометричних даних.

**Апробація отриманих результатів.** Висоцька О. О., Хукаленко Б.А. Аналіз використання сучасних методів біометричної автентифікації у менеджерах паролів, III Міжнародна науково-практична конференція “CURRENT TRENDS IN SCIENTIFIC RESEARCH DEVELOPMENT”, 17-19.10.2024, Бостон, США.: тези доповіді. – К., 2024. – С.137-143.

# РОЗДІЛ 1
# АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ З УПРАВЛІННЯ ПАРОЛІВ

## 1.1 Проблеми з досягнення безпеки інформаційних систем

В епоху цифрових технологій, коли інформація стала одним із найбільш цінних ресурсів, гарантія її безпеки набула виняткового значення. Інформаційні системи використовуються в усіх сферах людської діяльності: від бізнесу та фінансів до державного управління й освіти. Втрати або компрометація даних можуть мати серйозні наслідки для компаній, урядів та окремих людей, що робить проблему інформаційної безпеки однією з ключових у сучасному світі.

З розвитком технологій з'являються нові загрози та виклики. Кіберзлочинці постійно шукають способи обходити системи захисту, використовуючи як технічні недоліки, так і людський фактор. Соціальна інженерія, фішинг, шкідливе програмне забезпечення, атаки на мережі та витоки даних – це лише деякі з сучасних загроз, які стають все більш поширеними. В зв’язку з цим питання безпеки інформації постає дедалі гостріше.

Цінність інформації залежить від характеристик, якими вона володіє. Коли характеристика інформації змінюється, цінність цієї інформації або зростає, або, що частіше, зменшується [1].

Критичними характеристиками інформації є конфіденційність, цілісность та доступність, котрі часто називають триадою інформаційної безпеки (CIA: confidentiality, integrity, availability). Ці три елементи є основою побудови будь-якої системи безпеки:

- Конфіденційність (Confidentiality) означає захист інформації від несанкціонованого доступу. Метою конфіденційності є забезпечення того, щоб дані були доступні лише тим особам, яким вони призначені, і щоб сторонні не могли отримати доступ до цієї інформації. Конфіденційність особливо важлива для захисту персональних даних, фінансових даних, секретної інформації урядових або бізнес-структур.

- Цілісність (Integrity) передбачає збереження точності та повноти інформації. Це означає, що дані не повинні бути змінені або пошкоджені неавторизованими особами, а всі зміни повинні бути внесені лише з дозволу і зафіксовані. Цілісність важлива для забезпечення того, щоб дані були надійними і їх можна було використовувати для ухвалення рішень.
- Доступність (Availability) забезпечує своєчасний і надійний доступ до інформації для авторизованих користувачів. Це означає, що система або інформація повинні бути доступні в потрібний час і при необхідності, навіть у разі технічних збоїв або атак. Відсутність доступності може мати серйозні наслідки для бізнесу або організації, особливо в критичних інфраструктурах, таких як банки, лікарні, державні служби.

Ці три аспекти часто перебувають у певному балансі. Наприклад, занадто жорсткі заходи щодо забезпечення конфіденційності можуть вплинути на доступність інформації для авторизованих користувачів, а спроби підвищити доступність можуть знизити рівень захисту конфіденційності або цілісності. Тому завжди потрібно шукати оптимальні рішення, які враховують специфіку конкретної інформаційної системи.

Забезпечення конфіденційності, цілісності та доступності в комплексі дозволяє мінімізувати ризики втрати, викривлення або несанкціонованого доступу до інформації, що є основою для захисту даних у сучасному світі.

Інформаційна безпека охоплює різні аспекти, які забезпечують захист даних і інформаційних систем від різноманітних загроз. Залежно від природи заходів захисту, інформаційну безпеку можна розділити на три основні категорії: адміністративну, логічну та фізичну безпеку. Кожен із цих видів має свої специфічні методи, інструменти та принципи, які забезпечують комплексний підхід до захисту інформації.

Адміністративна безпека стосується управлінських аспектів інформаційної безпеки. Вона включає в себе політики, процедури, стандарти та практики, які визначають, як організація повинна поводитися з інформацією та

захищати її. Основною метою адміністративної безпеки є забезпечення належного управління ресурсами, а також визначення обов'язків і відповідальності співробітників щодо інформаційної безпеки.

Основні компоненти адміністративної безпеки:

- Політики безпеки – розробка та впровадження документів, що регламентують правила і процедури захисту інформації. Політики повинні бути зрозумілими, доступними та обов'язковими для виконання всіма співробітниками.
- Управління ризиками – оцінка потенційних загроз і вразливостей, а також розробка стратегій для їх зменшення. Це може включати аналіз бізнес–процесів, які містять інформаційні ризики.
- Навчання персоналу – регулярне навчання співробітників з питань інформаційної безпеки, яке допомагає підвищити обізнаність щодо загроз і методів захисту. Співробітники повинні знати, як діяти у разі інциденту безпеки.
- Контроль доступу – визначення ролей і прав доступу для різних категорій співробітників. Це може включати фізичний доступ до приміщень, а також доступ до інформаційних систем і даних.

Логічна безпека охоплює технічні засоби і методи захисту інформаційних систем. Вона стосується захисту даних від несанкціонованого доступу, змін і знищення шляхом використання програмного забезпечення, систем управління доступом та інших технологічних рішень. Логічна безпека є критично важливою в умовах, коли організації використовують комп'ютерні системи та мережі для зберігання та обробки даних.

Основні компоненти логічної безпеки:

- Автентифікація – процес перевірки особи або системи, що намагається отримати доступ до ресурсів. Може бути реалізовано через паролі, біометричні дані, токени або двофакторну автентифікацію.
- Шифрування – застосування криптографічних методів для захисту даних від несанкціонованого доступу. Шифрування даних під час

передачі (наприклад, HTTPS) та зберігання (шифровані бази даних) є важливими заходами захисту.

- Системи виявлення та запобігання вторгнень (IDS/IPS) – технології, які моніторять мережевий трафік і виявляють підозрілі дії або атаки. IDS виявляє загрози, тоді як IPS може автоматично вживати заходів для їх блокування.
- Антивірусне програмне забезпечення – використання ПЗ для виявлення та видалення шкідливих програм, які можуть загрожувати інформаційним системам.

Фізична безпека стосується захисту фізичних об'єктів і ресурсів організації, які містять інформацію або інформаційні системи. Вона включає заходи, які запобігають фізичному доступу несанкціонованих осіб до комп'ютерних систем, серверів, даних і самого приміщення. Фізична безпека є критично важливою, оскільки навіть найкращі технологічні заходи можуть бути знищені або скомпрометовані через фізичні втручання.

Основні компоненти фізичної безпеки:

- Контроль доступу до приміщень – використання карт доступу, ключів, кодів або біометричних систем для обмеження доступу до чутливих зон.
- Відеоспостереження – встановлення камер спостереження для моніторингу території та приміщень, що забезпечує контроль за фізичним доступом і може служити доказом у разі інциденту.
- Охорона – наявність охоронців або патрулів для забезпечення фізичної безпеки на території організації.
- Захист обладнання – заходи для запобігання фізичному знищенню або крадіжці обладнання, включаючи встановлення сигналізацій, використання замків та обмеження доступу до технічних приміщень.

Ці три види інформаційної безпеки тісно взаємопов'язані і доповнюють один одного. Наприклад, ефективна адміністративна безпека створює основи для логічних і фізичних заходів захисту, тоді як логічні та фізичні рішення реалізують політики та процедури адміністративної безпеки. Забезпечення

комплексного підходу до інформаційної безпеки є критично важливим для захисту даних і систем від різноманітних загроз.

Самі ж загрози інформаційним системам класифікуються як внутрішні та зовнішні. Внутрішні загрози походять від співробітників або користувачів системи, які можуть здійснювати дії, що призводять до порушення безпеки, як навмисно, так і через необережність. Вони включають зловживання доступом до інформації, помилки в конфігураціях систем, а також витоки даних через людський фактор. Зовнішні загрози виникають через дії сторонніх осіб або організацій, таких як хакери, конкуренти або організовані кіберзлочинні групи. Їхні дії можуть включати несанкціоновані атаки на інформаційні системи з метою крадіжки даних, шантажу або пошкодження інфраструктури.

### 1.2 Використання менеджера паролів задля вирішення проблем з інформаційної безпеки

У сучасному цифровому світі захист особистих даних став одним із найважливіших аспектів інформаційної безпеки. Зростання кількості онлайн-сервісів, соціальних мереж, електронних платіжних систем та інших платформ вимагало від користувачів створення численних облікових записів, кожен з яких вимагає надійного пароля. На жаль, більшість користувачів продовжують використовувати однакові або прості паролі, що суттєво знижує рівень безпеки їх облікових записів.

Аби детальніше розглянути проблему ненадійності паролів пересічних користувачів, звернемо увагу на останнє дослідження, проведене компанією NordPass [2].

NordPass у партнерстві зі сторонніми дослідниками проаналізували паролі з бази даних розміром 6,6 ТБ. Ці паролі були вкрадені різними зловмисними програмами, такими як Redline, Vidar, Taurus, Raccoon, Azorult і Cryptbot. Журнали зловмисного програмного забезпечення містять не лише паролі, а й веб-сайт джерела. Дослідники класифікували найпопулярніші паролі за країнами та поділилися статистичними результатами з NordPass. Далі, в таблиці

1.1 розглянемо найпопулярніші паролі у всьому світі в цілому та конкретно в Україні.

Таблиця 1.1

Найпопулярніші паролі за 2023 рік

| | Всі країни | | Україна | |
|---|---|---|---|---|
| | Пароль | Кількість | Пароль | Кількість |
| 1 | 123456 | 4,524,867 | admin | 11,140 |
| 2 | admin | 4,008,850 | 123456 | 8,766 |
| 3 | 12345678 | 1,371,152 | 123456789 | 3,736 |
| 4 | 123456789 | 1,213,047 | 12345678 | 2,626 |
| 5 | 1234 | 969,811 | 1234567890 | 2,224 |
| 6 | 12345 | 728,414 | 123123 | 2,065 |
| 7 | password | 710,321 | 111111 | 2,061 |
| 8 | 123 | 528,086 | qwerty | 1,654 |
| 9 | Aa123456 | 319,725 | 1234567 | 1,543 |
| 10 | 1234567890 | 302,709 | 1234qwer | 1,342 |

Як можна побачити з результатів дослідження представлених у табл. 1.1, переважна більшість паролів користувачів є надзвичайно простими та ненадійними, неспроможними надати адекватний рівень безпеки інформації. Вказаний час злому більшості з цих паролів, вказаних в оглянутому дослідженні складає менше однієї секунди.

Найскладніші для зламу паролі – випадкові символи та рядки. Однак, такі паролі вважаються надто складними для запам'ятовування в багатьох програмах. Маючи вибір, користувачі зазвичай створюють дуже спотворений розподіл вибору паролів, що відображає загальну семантику, яка полегшує запам'ятовування (наприклад, поєднання слова або послідовності чисел із особливим значенням) [3].

Для того, щоб створити надійний пароль, що зможе захистити користувача від взлому необхідно керуватись наступними принципами:

- Рекомендується використовувати паролі не менше 12–16 символів.

Довгі паролі значно складніше зламати за допомогою атаки методом перебору.

- Необхідно використовувати великі та малі літери, цифри та спеціальні символи (наприклад, !, @, #, $, %, ^, &, *). Це ускладнює паролі для зламу.
- Паролі, що містять тільки літери або лише цифри, значно легше зламати.
- Необхідно уникайти використання імен, дат народження, номерів телефонів або інших даних, які можуть бути легко вгадати.
- Такі паролі, як "123456", "password", "qwerty", або "abcdef", легко вгадуються. Вони часто є першими, які намагаються зламати зловмисники.
- Змінюйте регістр і вставляйте символи, щоб додати складності (наприклад, "Я люблю пити каву в 7 ранку!" –> "ЯЛпКв7р!").
- Не використовувати один і той же пароль для кількох облікових записів. Якщо один обліковий запис буде зламано, всі інші облікові записи, які використовують той самий пароль, також будуть під загрозою.

На жаль, як ми можемо бачити, створений за всіма правилами надійний пароль стане не лише надійним захистом від зловмисників, але й непосильним тягарем для пересічного користувача, особливо в разі використання різних паролів для всіх облікових засобів. Як результат, виходом з ситуації може послужити або архаїчний «метод» записування паролів на листочку, який сам по собі являться загрозою для безпеки; або ж, більш сучасний та актуальний метод – використання менеджера паролів.

Менеджери паролів – це програми, які генерують і зберігають складні паролі для різних облікових записів в Інтернеті [4]. Концепція менеджера паролів базується на принципі шифрування даних, що забезпечує безпечне зберігання конфіденційної інформації. Зазвичай менеджери паролів складаються з компонентів, приведених на рис. 1.1:

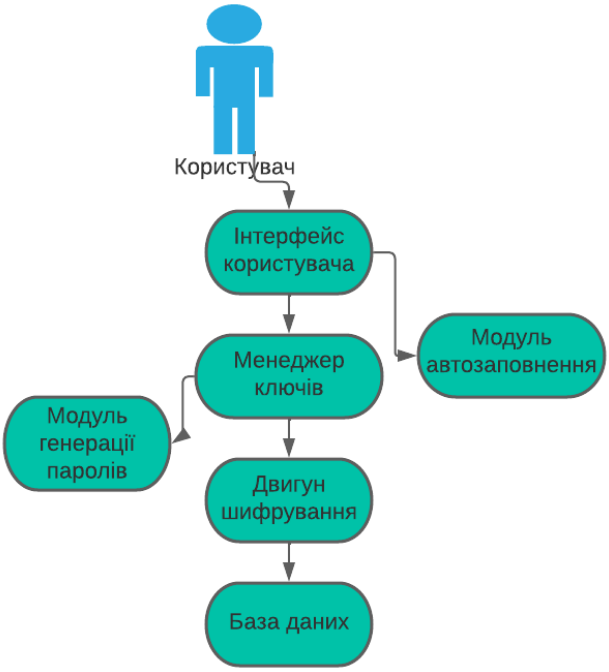


Рис. 1.1. Схема компонентів типового менеджеру паролів

1. Інтерфейс користувача. Слугує точкою взаємодії користувача з програмою. Забезпечує додавання, редагування, видалення записів про паролі. Простий та інтуітивно зрозумілий інтерфейс є важливим елементом для масового користувача.
2. Менеджер ключів. Відповідає за роботу з управління ключами шифрування. Забезпечує безпечне зберігання головного пароля користувача. Може використовувати додаткові методи автентифікації, такі як біометричні дані або двофакторна автентифікація.
3. Двигун шифрування. Забезпечує шифрування та дешифрування інформації в базі даних. Використовує сильні алгоритми шифрування (наприклад, AES) для захисту паролів. Ключ шифрування зазвичай походить від головного пароля користувача, який відомий тільки йому.
4. База даних. Зберігає всю інформацію про паролі користувача, включаючи самі паролі, URL-адреси, нотатки та іншу додаткову інформацію. Зазвичай використовується реляційна база даних або база даних NoSQL.

5. Модуль генерації паролів. Допомагає користувачам створювати сильні, унікальні паролі для кожного облікового запису. Використовує генератори випадкових чисел та набір символів для створення складних паролів.
6. Модуль автозаповнення. Взаємодія з веб-браузерами та додатками, дозволяючи користувачу автоматично заповнювати паролі з бази даних застосунку.

Менеджери паролів працюють за принципом зберігання даних у зашифрованому вигляді. Користувач вводить основний пароль (master password), який служить ключем для доступу до всіх збережених паролів. Введений пароль обробляється через хеш-функцію, що забезпечує подальше шифрування та зберігання в базі даних.

Хешування (рис. 1.2) є фундаментальною концепцією в криптографії, яка відіграє життєво важливу роль у забезпеченні цілісності, автентичності та безпеки даних. За своєю суттю, хешування включає процес перетворення вхідних даних довільного розміру в значення фіксованого розміру, відоме як хеш-значення або хеш-код, за допомогою застосування криптографічної хеш-функції [5]. Хеш-функції використовуються для перевірки цілісності даних і, що важливо для менеджерів паролів, для збереження головного паролю користувача у вигляді зашифрованого хешу, який складно відновити в початковий текст.


Рис. 1.2. Три базові компоненти процесу хешування

З огляду на збільшення кількості кібератак, актуальним постає питання підвищення інформаційної безпеки. З цією метою доцільно буде звернути увагу саме на менеджери паролів. Однією з основних причин використання менеджерів паролів є те, що вони зменшують ймовірність зламу паролів. У певному сенсі менеджери паролів схожі на захисні технології, такі як антишпигунське програмне забезпечення або антивірус. У цих випадках використання антивірусу зменшує ймовірність компрометації обчислювальних ресурсів [6]. До інших приводів для їх використання можемо віднести:

- Складність паролів. Менеджери паролів дозволяють створювати унікальні, складні паролі для кожного облікового запису, що ускладнює їх злом.
- Захист від фішингу. Багато менеджерів паролів мають вбудовані механізми захисту, що попереджають користувачів про потенційні фішингові сайти.
- Зручність. Використання менеджера паролів значно полегшує управління паролями, адже користувачеві потрібно пам'ятати лише один основний пароль.
- Аналіз безпеки. Деякі менеджери паролів пропонують функції аналізу безпеки паролів, що дозволяють виявити слабкі місця та рекомендують їх змінити.

Таким чином, можна сміливо сказати що менеджер паролів є ефективним засобом забезпечення інформаційної безпеки, що дозволяє користувачам зберігати паролі в безпеці, уникати фішингових атак і спростити процес управління обліковими записами в різних сервісах.

### 1.3 Аналіз сучасних методів автентифікації

Критично важливим елементом інформаційної безпеки, особливо в контексті використання менеджерів паролів, є автентифікація. Вона визначає, чи має користувач право на доступ до системи або ресурсів, і забезпечує перевірку

його особистості. Без ефективних методів автентифікації навіть найнадійніші менеджери паролів можуть стати вразливими до зловмисних атак.

В механізмах автентифікації, з метою їх подальшого порівняння та дослідження, виділяють 3 основні характеристики, зображені на рис. 1.3 [7]:

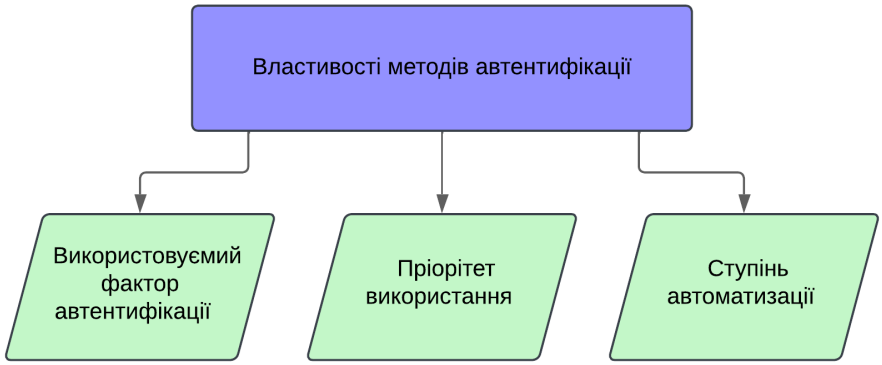


Рис. 1.3. Основні характеристики механізмів автентифікації

1. Ступінь автоматизації. Автоматизація може бути повною, або неповною. Мається на увазі автоматизація автентифікації з боку системи, а не користувача. Наприклад, система автентифікації більшості сайтів повністю автоматизована, а система автентифікації за допомогою домофона, автоматизована в неповному обсязі, тому що для автентифікації гостя необхідне втручання господаря.
2. Пріоритет використання. Ця характеристика визначає в якому порядку користувач користується даним методом автентифікації:
   - Основний метод автентифікації. Як видно з назви, цей метод використовується для штатного входу в систему. Найпоширеніший з них – вхід за паролем, який використовується в переважній більшості комп'ютерних систем.
   - Резервний метод автентифікації. У разі втрати пароля або е-токена, або взлому облікового запису в силу вступають резервні методи автентифікації. Найбільш поширені два методи: відповідь на «секретне питання» і відправка пароля на довірену поштову

скриньку, вказану при реєстрації.

- Механізм «останньої інстанції». Незважаючи на всі мінуси і слабкості таких механізмів резервного відновлення доступу до облікових записів, провідні інтернет компанії змушені ними користуватися. Це механізм, до якого вдаються в самих крайніх випадках, коли всі інші способи виявилися безсилі. В даний момент це означає звернення до адміністраторів інформаційних систем, або в спеціальні відділи підтримки клієнтів.

3. Використовуємий фактор автентифікації. Автентифікація являє собою процес порівняння інформації, що надається користувачем, з еталонною. Залежно від типу інформації її можна віднести до одного з трьох основних факторів, або до їх комбінації:
   - Фактор знання (Парольна автентифікація) – «те, що ти знаєш». Перший і найпоширеніший на даний момент механізм автентифікації, введення чогось, що відомо тільки користувачеві, наприклад, пароля або відповіді на секретне питання.
   - Матеріальний фактор (Апаратна автентифікація) – «те, чим ти володієш». В першу чергу під цим розуміються апаратно-програмні системи ідентифікації і автентифікації (СІА) або пристрої введення ідентифікаційних ознак. До складу СІА входять апаратні ідентифікатори, пристрої введення- виведення (зчитувачі, контактні пристрої, адаптери, роз'єми системної плати та ін.) і відповідне ПО.
   - Біофактор (Біометрична автентифікація) – «те, що є частиною тебе». Біометричні дані, для зняття яких, як правило, необхідні спеціальні програмно-апаратні засоби – так звані, біометричні сканери, які розрізняються за характером зчитувальних даних.

Беручи до уваги досліджені нами характеристики механізмів автентифікації, зазначені на рис. 1.3, розглянемо найпоширеніші сучасні технологічні рішення в цій сфері. Класифікацію технологій автентифікації зображено на рис. 1.4:

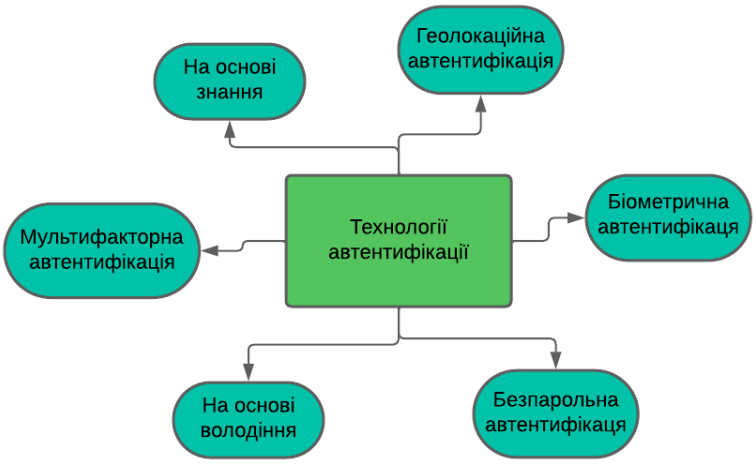


Рис. 1.4. Класифікація технологій автентифікації

1. Технології автентифікації на основі знання:
   - Хешування паролів. Паролі зберігаються в зашифрованому вигляді, зазвичай із застосуванням алгоритмів хешування (наприклад, SHA-256, bcrypt).
   - Сіль для паролів. Додавання унікального значення (модифікатора) до пароля перед хешуванням для запобігання атакам словників і таблицям райдужних хешів.
   - Капча. Тестування на "людяність", що запобігає автоматизованим атакам на паролі.
2. Технології автентифікації на основі володіння:
   - USB-ключі (апаратні токени). Використовуються апаратні засоби (наприклад, YubiKey, U2F), які генерують криптографічні ключі для підтвердження особи.
   - Мобільні токени (додатки). Мобільні додатки (Google Authenticator, Authy), що генерують одноразові паролі (OTP) на основі протоколу TOTP або HOTP.
   - Цифрові сертифікати. Використання цифрових сертифікатів (наприклад, X.509) для шифрування або підпису даних у процесі автентифікації.

3. Технології біометричної автентифікації:
    - Сканери відбитків пальців. Сенсори, що зчитують унікальні патерни відбитків для ідентифікації (використовуються на смартфонах, ноутбуках).
    - Розпізнавання обличчя. Використання камер і програмного забезпечення для аналізу біометричних характеристик обличчя (Apple Face ID, Windows Hello).
    - Сканування райдужної оболонки ока. Використання інфрачервоних сенсорів для детального аналізу структури ока.
4. Технології геолокаційної автентифікації:
    - GPS-автентифікація: Використання геолокаційних даних для підтвердження, що користувач перебуває в певному місці.
    - Wi-Fi ідентифікація. Автентифікація на основі підключених Wi-Fi мереж або близькості до відомих точок доступу.
    - IP-адреса. Використання інформації про IP-адресу для підтвердження особи (наприклад, дозволені доступи тільки з певних регіонів).
5. Технології мультифакторної автентифікації:
    - WebAuthn. Протокол для реалізації безпарольної автентифікації з підтримкою біометричних даних та криптографічних ключів.
    - FIDO2. Технологія, яка дозволяє використовувати апаратні та біометричні методи для автентифікації без паролів.
6. Технології безпарольної автентифікації:
    - Використання QR-кодів. Для автентифікації шляхом сканування унікального коду з мобільного пристрою.

Автентифікація запобігає доступу сторонніх осіб до збережених паролів, допомагає зменшити ймовірність крадіжки даних через фішингові атаки або використання зламаних паролів, а також підвищує обізнаність користувачів про необхідність використання складних і унікальних паролів для різних облікових записів. Розібравшись детальніше в різноманітних методах і технологіях

автентифікації та її позитивному впливі на захист систем інформації, можемо стверджувати що автентифікація є основою безпеки менеджерів паролів і критично важливим елементом у захисті особистих даних користувачів від загроз інформаційної безпеки.

### 1.4 Вибір оптимального методу автентифікації

Оскільки менеджери паролів, як інструменти для зберігання великої кількості цінної інформації, часто стають мішенню для зловмисників, традиційні методи автентифікації, такі як паролі та PIN-коди, хоча й зручні, мають свої обмеження: їх легко забути, підібрати або вкрасти.

З метою вибору оптимального методу автентифікації для застосунку менеджера паролів, створимо табл. 1.2, де проаналізуємо основні аспекти найпопулярніших варіантів:

Таблиця 1.2

Порівняння методів автентифікації

| Метод автентифікації | Переваги | Недоліки |
|---|---|---|
| Паролі | Простота впровадження, широке використання | Легко зламати, залежить від складності пароля, легко забути |
| Пін-коди | Легкість використання, швидка автентифікація | Низька безпека через коротку довжину, можуть бути вгадані або перехоплені |
| Токени | Високий рівень безпеки, унікальність кожного пристрою | Можуть бути втрачені або вкрадені, вимагають наявності додаткового пристрою |
| Цифрові сертифікати | Висока надійність, захист від підробки | Складність впровадження, потребує управління сертифікатами |
| Біометрія | Не потребує запам'ятовування, висока зручність і безпека, унікальність даних | Потребує спеціального обладнання, можливість помилкових спрацьовувань або відмов |

Користуючись даними з табл. 1.2, можемо дійти висновку що метод біометричної автентифікації є оптимальним вибором для використання в менеджері паролів завдяки кільком вагомим перевагам.

По-перше, біометричні дані, такі як відбитки пальців або геометрія обличчя, є унікальними для кожної людини, що значно підвищує рівень безпеки порівняно з традиційними паролями чи ПІН-кодами. Їх майже неможливо підробити або викрасти, що забезпечує захист від несанкціонованого доступу до конфіденційних даних, збережених у менеджері паролів.

Додатковою перевагою є зручність використання. Користувачеві не потрібно запам'ятовувати складні паролі або носити з собою фізичні токени. Автентифікація відбувається швидко й без зайвих зусиль – достатньо прикласти палець або подивитися в камеру. Такий підхід значно підвищує комфорт користувачів, особливо в умовах частого використання менеджера паролів для збереження й доступу до важливих облікових записів.

Окрім цього, біометричні характеристики неможливо втратити чи забути, на відміну від паролів або фізичних токенів. Відбитки пальців і риси обличчя завжди "під рукою", що робить біометрію надійнішим методом автентифікації. Важливо зазначити, що більшість сучасних мобільних пристроїв уже підтримують вбудовані технології біометричної автентифікації, такі як Apple Face ID, Touch ID та Windows Hello. Це спрощує інтеграцію біометрії в менеджер паролів, оскільки відповідні сенсори та програмне забезпечення вже наявні у багатьох смартфонах і планшетах.

Все це забезпечує найвищий рівень захисту та зручності для користувачів, що робить біометричну автентифікацію оптимальним вибором для реалізації у менеджері паролів. Тому, далі більш детально розглянемо біометричні технології.

Біометричний показник – це «вимірна фізіологічна та/або поведінкова риса, яку можна зафіксувати та згодом порівняти з іншими примірниками під час перевірки» [8]. Біометрична автентифікація базується на принципі цих показників людини для ідентифікації. Основна ідея полягає в тому, що кожна

особа має неповторні біологічні риси, які можна виміряти, зафіксувати та використовувати для доступу до систем або ресурсів.

Процес біометричної автентифікації зазвичай складається з двох основних етапів: реєстрації та автентифікації [9].

Реєстрація. На цьому етапі система збирає біометричні дані користувача і зберігає їх у захищеній базі даних або у вигляді хешованих/зашифрованих шаблонів. Наприклад, це можуть бути зображення відбитка пальця, риси обличчя, голосові зразки тощо. Після збору ці дані перетворюються у цифровий шаблон (відбиток), який використовуватиметься для порівняння на наступних етапах. Базу даних навчальних зразків доцільно періодично оновлювати в зв'язку зі змінами в біометричних характеристиках: з часом вони можуть змінюватися під впливом віку, травм, хвороб, хірургічних втручань тощо. До того ж, регулярне оновлення бази даних допомагає захистити систему від підробок біометричних даних а також зменшує ризик помилок при автентифікації.

Автентифікація. Коли користувач намагається отримати доступ до системи, він знову надає свої біометричні. Ці дані перетворюються у новий цифровий шаблон, який порівнюється із зареєстрованим шаблоном у базі даних. Якщо обидва шаблони достатньо збігаються, користувач вважається автентифікованим, і йому надається доступ.

Системи біометричної автентифікації складаються з чотирьох основних компонентів, зображених на рис. 1.5:

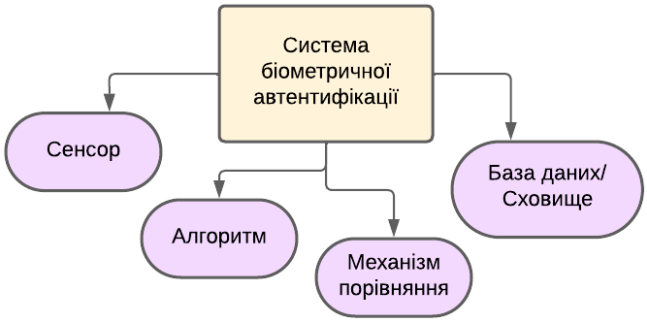


Рис. 1.5. Основні компоненти біометричної системи

- Сенсор: пристрій, що збирає біометричні дані (наприклад, сканер відбитків пальців, камера для розпізнавання обличчя або мікрофон для розпізнавання голосу).
- Алгоритм: програмне забезпечення, яке перетворює зібрані біометричні дані в цифровий шаблон.
- База даних або сховище: місце, де зберігаються зібрані біометричні шаблони користувачів для подальшого використання при автентифікації.
- Механізм порівняння: порівнює новий зразок біометричних даних з шаблоном, збереженим у системі, та визначає, чи є збіг достатнім для підтвердження особи.

### 1.5 Висновки до першого розділу

В цьому розділі ми проаналізували проблеми з досягнення безпеки інформаційних мереж. Зокрема, виділили три основні елементи побудови будь-якої системи безпеки: Конфіденційність, Цілісність та Доступність. До того ж, дослідили основні категорії інформаційної безпеки та їх компоненти.

Окрім цього, було проаналізовано дослідження з виявлення найпопулярніших паролів та принцип дії та структуру менеджера паролів.

Також було проведено аналіз сучасних методів автентифікації, зокрема, їх три основні характеристики: ступінь автоматизації, пріоритет використання та використовуваний фактор автентифікації.

Засновуючись на результатах проведеного аналізу, біометричну автентифікацію було обрано як оптимальний метод для використання в застосунку менеджера паролів.

В наступному розділі проведемо аналіз методів біометричної автентифікації з метою вибору найбільш підходящого варіанту в створюваному застосунку.

# РОЗДІЛ 2
# АНАЛІЗ МЕТОДІВ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ

## 2.1 Системи біометричної автентифікації

Біометричні системи для підтвердження особи застосовують унікальні фізичні дані користувача з метою його ідентифікації. Сама ж біометрична автентифікація полягає у перевірці особи через введення її особистих біометричних показників, які проходять обробку згідно з визначеним протоколом безпеки.

Використання біометричних систем доступу є дуже зручним для користувачів. На відміну від паролів чи фізичних носіїв, які можна втратити, викрасти або скопіювати, біометричні параметри людини завжди залишаються при ній, тож ризик їх збереження не виникає. Крім того, неможливо передати такі ідентифікатори іншим особам.

Зазвичай сучасні методи біометричної автентифікації поділяють на два класи, зображені на рис. 2.1:

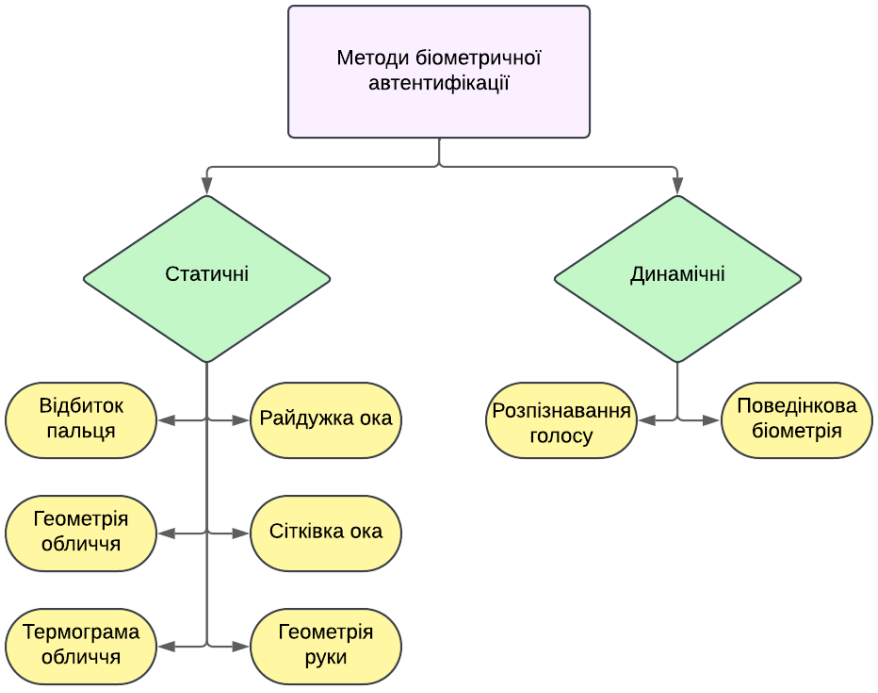


Рис. 2.1 Класифікація методів біометричної автентифікації

- Статичні методи – базуються на фізичних ознаках людини, які залишаються постійними або змінюються дуже повільно протягом усього життя. Ці особливості є індивідуальними для кожного, що робить їх викрадення або підробку практично неможливими.
- На відміну від них, Динамічні методи спираються на особливості поведінки людини, які хоч і залишаються індивідуальними, але можуть змінюватися з часом. Це свого роду "почерк" або "звички" користувача, наприклад, манера ходи, підпис або навіть ритм натискання клавіш на клавіатурі – усі ці характеристики є унікальними для кожної особи.

До статичних методів відносяться:

- Дактилоскопія – метод ідентифікації особи за відбитками пальців, протягом багатьох років використовується в різних сферах. Система розпізнавання відбитків пальців забезпечує хороший баланс безпеки, конфіденційності, зручності, і звітності [10]. Її застосування в менеджерах паролів є логічним кроком до підвищення рівня безпеки. Унікальність відбитків пальців, їхня стійкість до підробок та відносна простота використання роблять дактилоскопію привабливим варіантом для аутентифікації. Сучасні датчики дозволяють швидко і точно зчитувати відбитки пальців, що робить процес автентифікації зручним для користувача. Однак, дактилоскопія має і свої недоліки, такі як: проблеми з автентифікацією при пошкодженні пальців та можливість підробок при використанні низькоякісного обладнання.
- Ідентифікація за Геометрією обличчя має низку переваг, оскільки є основним біометричним показником, який люди використовують для розпізнавання один одного. Деякі з найдавніших ідентифікаційних маркерів, наприклад, портрети, використовують цей біометричний показник як шаблон автентифікації. Крім того, він добре сприймається і легко зрозумілий людям [11]. Перевагою використання розпізнавання геометрії обличчя є неінвазивність та зручність даного методу. Користувачеві не потрібно торкатися пристрою, достатньо просто

подивитися в камеру. Крім того, сучасні алгоритми розпізнавання обличчя стали значно впевненіше ідентифікувати користувача. З факторів, які обмежують застосування цієї технології в менеджерах паролів, можна виділити: якість зображення обличчя може значно вплинути на точність розпізнавання; недостатнє освітлення, наявність сторонніх предметів на обличчі, зміна зовнішнього вигляду з часом – все це може ускладнити процес ідентифікації

- Автентифікація за Термограмою обличчя використовує теплову карту, яка відображає унікальний розподіл температур на поверхні обличчя людини. Термограма отримується за допомогою інфрачервоної камери, яка фіксує невидиме теплове випромінювання. Цей метод дозволяє створити індивідуальний «тепловий відбиток» особи, який важко підробити, адже розподіл тепла на шкірі людини залежить від особливостей кровоносної системи та структури тканин. Технологія відрізняється високою надійністю, оскільки тепловий малюнок залишається відносно стабільним і не змінюється під впливом зовнішніх факторів, таких як макіяж або зміни освітлення. Даний метод має обмежене застосування по причині високої вартості необхідного обладнання та чутливість до змін температури навколишнього середовища, порте широко використовується в вузькоспеціалізованих сферах, потребуючих винятково високого рівню безпеки.
- Ірис-сканування – це високоточний біометричний метод ідентифікації, який базується на аналізі унікального візерунка райдужної оболонки ока. Дослідження та розробка технології розпізнавання райдужної оболонки ока сьогодні швидко розширюються в кількох десятках університетів і промислових дослідницьких центрів [12]. На відміну від відбитків пальців, які можуть змінюватися з часом або бути пошкодженими, візерунок райдужки є стабільним і практично неможливим для підробки. Іншою перевагою сканування райдужки є його безконтактність. Проте, головним недоліком методу є висока

вартість обладнання. Спеціалізовані сканери райдужки є досить дорогими, що обмежує їх використання для пересічного користувача менеджера паролів. Крім того, якість зображення райдужки має бути дуже високою для забезпечення точної ідентифікації. Це може бути проблематично в умовах недостатнього освітлення або при рухах очей.

- Автентифікація за сітківкою ока – це метод, що базується на аналізі унікального малюнка кровоносних судин, розташованих на сітківці. Для отримання такого зображення використовується інфрачервоне освітлення, яке допомагає зчитувати цей малюнок без впливу зовнішнього освітлення. Сітківка є стабільною анатомічною характеристикою, яка майже не змінюється з часом, що забезпечує високий рівень захисту та точності ідентифікації. Метод забезпечує високу надійність, оскільки судинний малюнок сітківки неможливо підробити або відтворити, що робить автентифікацію за сітківкою одним із найзахищеніших методів. Проте його застосування обмежене через потребу у спеціальному обладнанні, високу вартість технології, та процес зчитування даних, за якого необхіднітсь зафіксувати око на невеликій відстані від сканеру та залишатися абсолютно нерухомим протягом декількох хвилин стає причиною значного дискомфорту у користувачів.
- Автентифікація за геометрією руки є методом біометричної ідентифікації, який використовує індивідуальні особливості форми та будови людської руки. Ця технологія базується на аналізі параметрів, таких як довжина і ширина пальців, розмір долоні, а також співвідношення між різними частинами руки. За допомогою спеціального сканера зчитуються тривимірні дані про форму руки, що потім порівнюються з раніше збереженими зразками в базі даних. Процес автентифікації зазвичай проходить швидко і не потребує високої точності розташування руки, що забезпечує зручність у використанні. Оскільки форма руки є відносно стабільною в дорослому

віці, така біометрична ознака може бути використана протягом тривалого періоду без потреби в оновленні даних. Цей метод широко застосовується там, де необхідний середній рівень безпеки, наприклад, у підприємствах для контролю доступу співробітників.

До динамічних методів відносяться:

- Розпізнавання голосу – технологія біометричної ідентифікації, що набуває все більшої популярності у сфері інформаційної безпеки. Голос кожної людини має унікальні характеристики, такі як тембр, інтонація, швидкість мовлення та особливості вимови окремих звуків. Ці характеристики використовуються для створення голосового відбитка, який може бути використаний для ідентифікації особистості. Однією з головних переваг розпізнавання голосу є його зручність. Для аутентифікації користувачеві достатньо просто сказати кілька слів або фраз. Однак, розпізнавання голосу має і свої недоліки. Якість розпізнавання може погіршуватися через шум, зміну тембру голосу (наприклад, при застуді), використання гарнітури або диктофона. Крім того, існують ризики підробки голосу за допомогою записів або спеціальних програм.
- Поведінкова біометрія – це інноваційний підхід до біометричної ідентифікації, який аналізує унікальні шаблони поведінки користувача. На відміну від традиційних методів, таких як відбитки пальців чи розпізнавання обличчя, поведінкова біометрія фокусується на тому, як користувач взаємодіє з пристроєм. Одним з ключових переваг поведінкової біометрії є її неінвазивність. Для ідентифікації користувача не потрібно збирати біологічні зразки або виконувати спеціальні дії. Достатньо просто використовувати пристрій так, як зазвичай. Системи поведінкової біометрії аналізують різноманітні параметри, такі як: швидкість набору тексту, тиск на клавіші, рухи миші і т.ін. Однак, поведінкова біометрія має і свої обмеження. Точність ідентифікації може знижуватися під впливом зовнішніх факторів, таких

як стрес, втома або хвороба. До того ж, існує ризик, що зловмисник зможе підробити поведінковий профіль користувача, спостерігаючи за його діями.

## 2.2 Статичні методи біометричної автентифікації

Як ми зазначили раніше, до статичних методів відносяться ті, що використовують незмінні з часом характеристики людського тіла. Подібні методи відрізняються залежно від конкретних фізичних ознак, на яких вони базуються. Розгляньмо детальніше найпопулярніші методи, що зазвичай застосовуються для побудови систем автентифікації.

### 2.2.1 Ідентифікація за відбитком пальця

Одним з найбільш поширених методів біометричної ідентифікації є визначення особи за відбитком пальця. Один з підходів реалізації цього методу полягає у використанні унікальних точок на зображенні відбитка – мінюцій. Мінюції включають місця розгалуження, де гребінь ділиться на дві лінії, та закінчення гребенів, де лінія завершується.

Такий метод виявляється стійким до шумів і дефектів зображень. Проте під впливом різних перешкод, наприклад, пошкоджень шкіри або умов сканування, ці характеристики можуть змінюватися. Це може негативно вплинути на точність ідентифікації, оскільки зміни в таких точках впливають на топологічні та інші особливості зображення.

На першому етапі проводиться обробка зображення для покращення його якості, усунення шумів та полегшення подальшого виявлення мінюцій. Основні кроки цього етапу включають:

Нормалізація зображення – дозволяє вирівняти яскравість і контрастність для зменшення впливу зовнішніх чинників, таких як освітлення чи неоднорідність [13]. Нормалізація може бути виконана за формулою:

$$I` = \frac{I - \mu}{\sigma} \times \sigma_0 + \mu_0, \qquad (2.1)$$

де I – значення пікселя;

μ – середнє значення яскравості в області;

σ -стандартне відхилення;

$\mu_0$ і $\sigma_0$ – бажані середнє значення та відхилення.

Підсилення контурів і зменшення шуму. Для підсилення гребенів та приглушення шумів часто використовують Габоровий фільтр, що підкреслює текстуру, роблячи її більш помітною і чіткою:

$$g(x, y;\ \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 x y'^2}{2\sigma^2}\right) \times \cos\left(2\pi \frac{x'}{\lambda} + \psi\right), \qquad (2.2)$$

де: λ – довжина хвилі (визначає частоту синусоїдальної компоненти);

θ – орієнтація фільтра (кут, на який повернуто синусоїдальну хвилю);

ψ – фазовий зсув синусоїдальної компоненти;

σ – ширина гаусіани (визначає просторову масштабність фільтра);

γ – коефіцієнт аспекту (визначає співвідношення ширини й висоти гаусіани).

Скелетизація. Після фільтрації зображення проводиться скелетизація – це процес перетворення всіх ліній гребенів на однопіксельну ширину. Це зменшує кількість інформації, що необхідна для аналізу, залишаючи при цьому всі ключові характеристики зображення.

Наступний етап – виявлення мінцюцій. Він включає виділення точок розгалужень і закінчень гребенів. В процесі виявлення мінюцій ідентифікують позиції точок з координатами (x,y) та орієнтацією кожного гребеня θ, що визначає напрямок гребеня в кожній мінюції.

Процес виявлення мінюцій зазвичай включає наступне:

- Аналіз структури скелетизованого зображення. На цьому кроці зображення аналізується, щоб знайти всі точки, де лінії або розгалужуються, або закінчуються. Ці точки позначаються як мінюції.
- Видалення зайвих мінюцій. Іноді після аналізу з’являються "помилкові" мінюції, які виникають через шум або пошкодження зображення. Їх видаляють або ігнорують для забезпечення максимальної точності.

Після виявлення мінюцій з кожної з них створюється так званий дескриптор – набір характеристик, що описує кожну мінюцію. Дескриптор мінюцій можна представити у вигляді вектора:

$$M_i = (x_i, y_i, \theta_i), \tag{2.3}$$

де $x_i, y_i$ – координати мінюції;

$\theta_i$ – орієнтація гребеня.

Дескриптор мінюцій є унікальним для кожного відбитка пальця і забезпечує точне порівняння з еталонним зразком.

Основний процес ідентифікації полягає в порівнянні мінюцій тестового зображення з мінюціями еталонного. Оскільки відбиток може бути трохи повернутим чи зміщеним, цей етап включає нормалізацію положення, щоб уникнути помилок через невідповідність положень точок.

Нормалізація зображення**.** Для вирівнювання мінюцій використовується трансляція та обертання координат. Цю трансформацію можна виразити через матрицю:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{2.4}$$

де α – кут обертання

$t_x$ і $t_\text{y}$ – зміщення по осях.

Обчислення відповідності. Відстань між парами мінюцій обчислюється за формулою:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\ , \tag{2.5}$$

Також визначається різниця орієнтацій:

$$\Delta\theta = \left|\theta_i - \theta_j\right|. \tag{2.6}$$

Якщо значення d та $\Delta\theta$ не перевищують заданих порогів, мінюції вважаються відповідними.

Остаточна оцінка відповідності обчислюється за допомогою коефіцієнта схожості:

$$S = \frac{\text{Кількість відповідних мінюцій}}{\text{Загальна кількість мінюцій}}. \tag{2.7}$$

Значення S свідчить про рівень відповідності між двома відбитками: чим вищий цей коефіцієнт, тим більша ймовірність того, що обидва зразки належать одній особі. Якщо S перевищує встановлений поріг, відбитки вважаються ідентичними.

### 2.2.2 Ідентифікація за геометрією обличчя

Алгоритм ідентифікації за геометрією обличчя базується на аналізі унікальних характеристик обличчя, таких як відстані між ключовими точками (очі, ніс, рот, підборіддя). Ці точки створюють індивідуальний «каркас» для кожної особи, який допомагає розпізнавати людину.

Виявлення обличчя: спочатку система визначає положення обличчя в зображенні. Для цього часто застосовують методи на основі каскадних

класифікаторів (наприклад, алгоритм Віоли-Джонса), які знаходять контури обличчя за певними ознаками яскравості та контрастності.

Алгоритм Віоли-Джонса працює в три основних етапи:

1. Інтегральне зображення – дозволяє швидко обчислювати суми пікселів у прямокутній області. Інтегральне зображення I у точці (x,y) обчислюється як сума всіх пікселів вище та лівіше від цієї точки:

$$I(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} f(x',y'), \tag{2.8}$$

де $f(x',y')$ – інтенсивність пікселя в початковому зображенні.

2. Каскадний класифікатор на основі хаар-подібних ознак: алгоритм використовує набір прямокутних шаблонів (ознаки Хаара), що оцінюють зміни інтенсивності в різних частинах обличчя. Наприклад, брови темніші, ніж верхня частина чола. Ці шаблони формалізуються через різницю між сумами пікселів у двох сусідніх областях:

$$H = \sum_{i \in R_1} I(i) - \sum_{i \in R_2} I(i), \tag{2.9}$$

де $R_1$ та $R_2$ – області, що аналізуються.

3. Каскадний підхід: замість обробки всього зображення алгоритм аналізує лише області, які можуть містити обличчя, за допомогою багаторівневих класифікаторів. Це забезпечує високу швидкість роботи.

Після виявлення обличчя ключові точки (зазвичай, це центр кожного ока, кінчик носа, кути рота, скроні, підборіддя) визначаються з використанням методів машинного навчання, таких як глибокі нейронні мережі або локалізація орієнтирів :

1. Попередня обробка: зображення нормалізується, зокрема, коригується масштаб, нахил і яскравість.
2. Ключові точки: нехай $P = \{p_1, p_2, \dots, p_n\}$, де $p_i = (x_i, y_i), -$ набір координат ключових точок, отриманих на основі нейронної мережі або статистичних моделей.
3. Метод PCA (аналіз головних компонент): Якщо точки обличчя є векторами високої розмірності, для зменшення їх розмірності та виділення основних ознак можна застосувати PCA:

$$Z = W^T * (P - \bar{P}), \tag{2.10}$$

де W – матриця головних компонент;

$\overline{P}$ – середнє значення координат ключових точок.

Після визначення координат ключових точок обчислюються відстані та кути між цими точками. Набір цих характеристик формує геометричний профіль обличчя.

Якщо $P_i = (x_i, y_i)\ i\ P_j = (x_j, y_j)$ – координати двох ключових точок обличчя, відстань $D_{ij}$ між цими точками обчислюється за формулою:

$$D_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \tag{2.11}$$

Це дозволяє отримати унікальні відстані, наприклад, між очима, між носом і ротом або між підборіддям і очима.

Для визначення нахилу між трьома точками (наприклад, кут між носом і обома очима) дикористову.nm формулу косинусів. Нехай три точки мають координати $A = (x_1, y_1), B = (x_2, y_2), C = (x_3, y_3)$. Тоді кут θ при точці В між відрізками АВ та ВС можна знайти за формулою:

$$\cos\theta = \frac{(x_2 - x_1)(x_3 - x_2) + (y_2 - y_1)(y_3 - y_2)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} * \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}}, \tag{2.12}$$

Цей кут є важливим для ідентифікації, оскільки він враховує унікальні нахили між рисами обличчя.

Для підвищення точності можна виконати нормалізацію отриманих значень відстаней і кутів. Це дозволяє зменшити вплив різних масштабів зображень. Один зі способів нормалізації – поділити всі відстані на відстань між певними контрольними точками, наприклад, між очима:

$$D_{ij}^{'} = \frac{D_{ij}}{D_{\text{між очима}}}, \tag{2.13}$$

Ці характеристики об'єднуються в єдиний вектор $F = [D_{ij}^{'}, \cos\theta]$, який є шаблоном обличчя.

Шаблон порівнюється з існуючими шаблонами в базі даних за допомогою метрик схожості.

Евклідова відстань:

$$S = \sqrt{\sum_{k=1}^{n} (F_k - F_k^{'})^2}, \tag{2.14}$$

де $F_k$ і $F_k^{'}$ – шаблони порівнюваних облич.

Косинусна схожість:

$$S = \frac{\sum_{k=1}^{n} F_k * F_k'}{\sqrt{\sum_{k=1}^{n} F_k^2} * \sqrt{\sum_{k=1}^{n} F_k'^2}}, \tag{2.15}$$

Чим менше значення S, тим більше схожість між шаблонами.

### 2.2.3 Ідентифікація за термограмою обличчя

Ідентифікація за термограмою обличчя базується на аналізі інфрачервоного випромінювання, яке генерується шкірою людини. Термограма

є унікальним «тепловим підписом», що залежить від фізіологічних характеристик кровоносної системи, структури обличчя та температури шкіри.

Цей метод працює навіть у темряві та є менш чутливим до зміни пози обличчя, оскільки тепло не залежить від освітлення.

Алгоритм поділяється на ключові етапи:

1. Збір термограм

Система використовує інфрачервоні камери для зйомки обличчя. Інфрачервоний сенсор перетворює теплове випромінювання в цифрове зображення, де кожен піксель відповідає певній температурі T(x,y).

Термограму можна уявити як функцію:

$$I_T(x,y) = f\big(T(x,y)\big), \tag{2.16}$$

де T(x,y) – температура точки з координатами (x,y);

$I_T(x,y)$ – інтенсивність пікселя.

2. Попередня обробка зображення

Щоб зменшити вплив змін температури навколишнього середовища, проводять нормалізацію:

$$T'(x,y) = \frac{T(x,y) - T_{min}}{T_{max} - T_{min}}, \tag{2.17}$$

де $T_{max}$ і $T_{min}$ – мінімальна та максимальна температура на зображенні.

Для покращення якості термограми застосовують фільтри, наприклад, Гауссовий фільтр:

$$I'_T(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I_T(x+i, y+j) * G9i,j), \tag{2.18}$$

де G(i,j) – ядро Гаусса.

Для відокремлення обличчя від фону застосовують методи порогової сегментації:

$$S(x,y) = \begin{cases} 1, & T'(x,y) > T_{\text{поріг}} \\ 0, & \text{інакше} \end{cases}, \quad (2.19)$$

3. Виділення ознак

Після сегментації виділяють унікальні ознаки термограми, які є основою для ідентифікації.

Ключові теплові точки – визначаються як локальні максимуми температури:

$$\nabla T(x,y) = 0 \text{ і } \Delta T(x,y) < 0, \quad (2.20)$$

де ΔT(x,y) – лапласіан температури.

Теплові патерни – аналізуються області підвищеного теплового випромінювання, наприклад, навколо очей, носа або рота.

Геометрія теплових зон – визначаються розміри та відстані між тепловими зонами. Якщо $P_i = (x_i, y_i)$ *і* $P_j = (x_j, y_j)$ – координати центрів теплових зон, то відстань між ними:

$$D_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \quad (2.21)$$

4. Створення теплового шаблону

Тепловий шаблон представляє сукупність ознак:

- координати ключових точок $P = \{(x_1, y_1), (x_2, y_2), \dots\}$;
- температури цих точок $T_P = \{T_1, T_2, \dots\}$;
- відстані між тепловими зонами $D_{ij}$;
- текстурні характеристики (наприклад, ентропія температурних значень).

Усе це зберігається у вигляді вектора ознак:

$$F = \left[P, T_P, D_{ij}, \text{текстурні ознаки}\right], \tag{2.22}$$

5. Порівняння з базою даних

Шаблон порівнюється з шаблонами в базі даних за допомогою метрик схожості.

Евклідова відстань:

$$S = \sqrt{\sum_{k=1}^{n} (F_k - F_k')^2}, \tag{2.23}$$

де $F_k$ і $F_k'$ – вектори шаблонів термограм.

Кореляційна міра:

$$\rho = \frac{\sum_{k=1}^{n} (F_k - \bar{F})(F_k` - \bar{F}`)}{\sqrt{\sum_{k=1}^{n} (F_k - \bar{F})^2} * \sqrt{\sum_{k=1}^{n} (F_k` - \bar{F}`)^2}}, \tag{2.24}$$

де $\bar{F}$ і $\bar{F}`$ – середні значення компонент векторів.

Чим більша кореляція ρ, тим більша ймовірність збігу.

### 2.2.4 Ідентифікація за райдужною оболонкою ока

Ідентифікація за райдужною оболонкою базується на аналізі унікальних текстурних та структурних особливостей, які присутні в райдужній оболонці кожної людини.

Основні етапи алгоритму ідентифікації:

1. Захоплення зображення

Використовується спеціальна камера з інфрачервоним підсвічуванням для зйомки райдужної оболонки ока. Інфрачервоне випромінювання допомагає

мінімізувати вплив освітлення та підкреслити дрібні текстурні особливості оболонки.

Зображення райдужної оболонки представляється як матриця інтенсивності:

$$I(x, y) = f(x, y), \tag{2.25}$$

де I(x,y) – інтенсивність пікселя на координатах (x,y).

2. Локалізація райдужної оболонки

Зображення ока містить райдужну оболонку, зіницю, склеру та повіки. Для подальшого аналізу необхідно виділити межі райдужної оболонки.

Для визначення меж зіниці використовується метод Хафа для пошуку кола:

$$(x_c, y_c, r) = \arg\max \sum_{\theta} I(x_c + r\cos\theta, y_c + r\sin\theta), \tag{2.26}$$

де $(x_c, y_c)$ – центр зіниці,

r – радіус.

Задля визначення зовнішнього контуру райдужної оболонки аналогічно застосовується метод Хафа для пошуку більшого кола, яке визначає межі райдужної оболонки.

Сегментація повік – використовуються порогові методи або сплайн-апроксимація для видалення областей, що перекривають оболонку.

3. Нормалізація райдужної оболонки

Райдужна оболонка трансформується у нормалізовану прямокутну область для спрощення аналізу.

Процес описується як полярна трансформація:

$$I(r, \theta) = I(x_c + r\cos\theta, y_c + r\sin\theta), \tag{2.27}$$

де r – відстань від центру зіниці, θ – кут у полярній системі координат.

Нормалізоване зображення має розмірність R × Θ, де R – кількість радіальних сегментів, а Θ – кількість кутових сегментів.

4. Виділення ознак

На цьому етапі виділяють унікальні характеристики райдужної оболонки.

Для аналізу текстурних ознак застосовуються Габорові фільтри:

$$G(x, y;\ \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 x y'^2}{2\sigma^2}\right) \times \cos\left(2\pi \frac{x'}{\lambda} +\ \psi\right), \qquad (2.28)$$

де $x` = x\cos\theta + y\sin\theta$,

$y` = -x\sin\theta + y\cos\theta$

Текстурні ознаки кодуються у вигляді вектора, який представляє бітовий шаблон райдужної оболонки:

$$C = [c_1, c_2, \dots, c_n], \qquad (2.29)$$

де $c_i \in \{0,1\}$ – результати бінаризації відповіді фільтра.

5. Створення шаблону

Шаблон райдужної оболонки представляється як бітова матриця:

$$B = \{b_{ij}\}, b \in \{0,1\}, \qquad (2.30)$$

де $b_{ij}$ відповідає результату обробки пікселя з координатами (i,j).

Шаблон також включає "маску", яка вказує на області, що були затемнені повіками чи іншими перешкодами.

6. Порівняння шаблонів

Для порівняння шаблонів використовується метод Хеммінгової відстані:

$$HD = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(B_{ij} \oplus B_{ij}^{`}\right) * M_{ij} * M_{ij}^{`}}{\sum_{i=1}^{n}\sum_{j=1}^{m} M_{ij} * M_{ij}^{`}}, \qquad (2.31)$$

де B і B` – бітові шаблони;

M і M` – маски;

$\oplus$ – виключне "АБО" (XOR).

Якщо HD менше заданого порогу, шаблони вважаються відповідними.

## 2.3 Динамічні методи біометричної автентифікації

Як було зазначено раніше, динамічні методи використовують характеристики, що можуть змінюватися з плином життя людини. Подібно до статичних методів, ці методи класифікуються залежно від типу характеристик, з якими вони працюють. Тепер розглянемо найбільш популярні методи, які найбільш поширеними при створенні систем автентифікації.

### 2.3.1 Ідентифікація за клавіатурним почерком

Клавіатурний почерк – це динамічна (поведінкова) характеристика людини, яка може трохи змінюватися за декількох причин. Людина може поспішати кудись, може відволіктися на якісь зовнішні обставини, може захворіти – при цьому вона втрачає уважність і це позначається на її почерку, тобто відбувається відхилення почерку від його норми для даної людини. Аналіз цих відхилень є корисним під час виконання моніторингу за роботою співробітників підприємства або під час прийому на роботу. Якщо ж почерк змінився істотно, тоді це, як правило, свідчить про несанкціоновану зміну користувача КС [13].

Відомий метод визначення законів розподілу випадкових величин, який є апаратурним аналізом випадкових процесів, використовують еталонні гаусові сигнали. Цей метод полягає в наступному:

1. Формуються еталонні сигнали гауссовського виду, які описуються функціями щільності ймовірності $W_n(x)$.
2. Для кожних допоміжних сигналів $x_m(t)$ та і-ої реалізації аналізованого процесу $x_i(t)$ формуються сигнали подібності за формулою:

$$S_{ni} = f(d_{ni}) = |1 + d_{ni}| - 1, n = 1,2 \dots N, \quad (2.32)$$

де $d_{ni}$ – відстань між вибірками еталонних процесів і аналізованого процесу.

3. і-та реалізація досліджуваного процесу включається в деяку підсукупність $X_n$ після порівняння вихідних сигналів із заданим пороговим рівнем $h_n$.

Результатом порівняння сигналів подібності між собою є поділ генеральної сукупності реалізацій I розглянутого процесу на під-сукупності реалізацій $I_n, n = 1,2, \dots, N$. Так як кожна n-на під-сукупність розглянутого процесу характеризується деяким ядром, тобто багатовимірною щільністю розподілу ймовірностей n-го допоміжного процесу $W_n(x)$, то багатовимірні закони розподілу заданого процесу описуються сумішами щільності розподілу із наступними ваговими коефіцієнтами [14]:

$$q_n = \frac{I_n}{1, n = 1,2, \dots, N}, \quad (2.33)$$

Для реалізації методу поділу суміші сигналів гауссової форми, використовуючи еталонні сигнали, застосовують статистичний аналіз. Як еталонні сигнали обираються тимчасові параметри натискання та відпускання клавіш. Формування еталону здійснюється шляхом вимірювання інтервалів між натисканням і відпусканням клавіш користувачами, які вводять певний текст на клавіатурі. Після того, як зібрано набір характеристик клавіатурного почерку, проводиться їх очищення від аномальних значень, шуму та викидів. Очищені дані використовуються для обчислення математичних очікувань, які потім служать еталонними значеннями для подальшої ідентифікації клавіатурного почерку.

### 2.3.2 Ідентифікація за голосовими характеристиками

Ідентифікація за голосовими характеристиками (біометрія голосу) базується на аналізі фізичних і динамічних параметрів, властивих людському

голосу. Цей метод використовує різні акустичні ознаки, такі як частота, тембр, інтенсивність звуку, ритм мовлення та інші характеристики, що є унікальними для кожної особи. Алгоритм ідентифікації за голосом складається з кількох основних етапів, які включають запис голосу, обробку та порівняння з шаблоном.

1. Запис голосу

Першим етапом є запис голосу користувача за допомогою мікрофона або іншого пристрою для аудіозапису. Запис здійснюється на певний період часу (наприклад, кілька секунд), під час якого користувач вимовляє заздалегідь визначене слово або фразу.

2. Обробка сигналу

Отриманий аудіосигнал обробляється для виділення важливих характеристик голосу. Основні етапи обробки:

Перетворення сигналу у частотну область (Fourier перетворення) – аудіосигнал перетворюється з часового в частотний спектр, що дозволяє виділити основні акустичні компоненти.

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt, \tag{2.34}$$

де X(f) – спектр сигналу;

x(t) – аудіосигнал;

f – частота.

Виділення мел-частотних кепстральних коефіцієнтів (MFCC): одним з найбільш популярних методів для аналізу голосу є обчислення MFCC. Для цього спектр сигналу розбивається на кілька частотних смуг, з яких потім обчислюються параметри, що описують акустичну енергію в кожній смузі.

$$MFCC_k = \sum_{n=1}^{N} \log(|X_n|) * \omega_{kn}, \tag{2.35}$$

де $MFCC_k$ – коефіцієнти для k-ї смуги, $X_n$ – спектральні компоненти сигналу, $\omega_{kn}$ – вага для кожної частоти.

3. Створення шаблону голосу

Після обробки сигналу виділяються характеристики голосу, які використовуються для створення шаблону. Для кожної людини створюється індивідуальний шаблон, що включає в себе MFCC та інші акустичні параметри (наприклад, основну частоту голосу, інтенсивність звуку, тривалість звуків).

Шаблон голосу можна представити у вигляді вектора:

$$T = \{ MFCC_1, MFCC_2, \dots, MFCC_m \}, \tag{2.36}$$

де T – шаблон;

m – кількість характеристик, що використовуються для ідентифікації.

4. Порівняння з базою даних

Ідентифікація здійснюється шляхом порівняння шаблону голосу користувача з шаблонами, що зберігаються в базі даних. Для цього використовуються різні методи порівняння, зокрема методи класифікації або обчислення відстані між шаблонами.

Евклідова відстань:

$$D(T, T`) = \sqrt{\sum_{i=1}^{m} (T_i - T`_i)^2}, \tag{2.37}$$

де T – шаблон голосу користувача, T′ – шаблон у базі даних.

Інший метод порівняння, що застосовується для голосової ідентифікації, полягає в побудові Гауссових сумішей, де кожен шаблон голосу моделюється як суміш декількох нормальних розподілів:

$$P(x|\lambda) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k), \quad (2.38)$$

де $P(x|\lambda)$ – мовірність належності до шаблону λ;

$\pi_k$ – вага k-ї компоненти;

$\mu_k$ і $\Sigma_k$ – середнє і коваріація для k-ї компоненти;

$N(x|\mu_k, \Sigma_k)$ – нормальний розподіл.

5. Прийняття рішення

Після порівняння шаблонів система приймає рішення. Якщо відстань між шаблонами є меншою за заданий поріг ∈, то ідентифікація вважається успішною:

$$D(T, T`) \leq \in, \quad (2.39)$$

де D(T,T′) – відстань між шаблонами;

∈ – поріг, що визначає максимальну допустиму відстань для успішної ідентифікації.

## 2.4 Обґрунтування доцільності побудови застосунку менеджеру паролів на основі біометричної автентифікації саме на базі методу ідентифікації за відбитком пальця.

Для того, щоб обрати найефективнішу біометричної технології розпізнавання необхідно провести аналіз основних методів, які є найпоширенішими у цій сфері. Оскільки статичні методи демонструють більшу стабільність і простоту реалізації в порівнянні з динамічними, доцільно зосередитися саме на них.

Під час розробки системи біометричної автентифікації користувачів першочерговою вимогою є забезпечення високої точності розпізнавання особи. Проте значну роль також відіграють такі фактори, як зручність у використанні, швидкодія, розповсюдженість та вартість експлуатації.

Для оцінки різних методів біометричної ідентифікації застосовують різноманітні показники, зокрема помилку першого та другого роду. Помилка першого роду (ППР) виникає тоді, коли система помилково відхиляє коректний результат ідентифікації. Натомість помилка другого роду (ПДР) трапляється, якщо система хибно приймає негативний результат як позитивний.

Ще одним важливим аспектом для порівняння методів є їхня стійкість до підробки автентифікаційних зразків. Цей параметр відіграє критичну роль, адже навіть найбільш точна система, здатна виявляти найменші відмінності між зразками, може втратити свою ефективність, якщо підробка автентифікаційних даних є простою і доступною.

Стабільність характеристик у часі є важливим аспектом для забезпечення довгострокової релевантності еталонних зразків у системі ідентифікації. Це означає, що навіть якщо характеристики користувача незначно змінюються, система повинна мати можливість правильно ідентифікувати його за попередньо отриманими зразками.

Велику роль у стабільності системи відіграє також вплив зовнішніх факторів на процес розпізнавання. Ці фактори включають рівень освітлення, температуру навколишнього середовища, кут падіння світла, рівень пилу, склад повітря та інші умови. За умови мінімальної схильності системи до впливу зовнішніх факторів досягається збільшення її точності та надійності.

Ще одним критичним параметром є час, необхідний для ідентифікації особи. Швидкість розпізнавання залежить від часу, необхідного для зчитування необхідних ознак, кількості використовуваних для порівняння шаблонів, обчислювальної потужності та складності алгоритмів, що використовуються для обробки даних.

Додатковою перевагою є можливість безконтактної ідентифікації, яка забезпечує вищий рівень гігієнічності. Це особливо актуально для громадських або багатокористувацьких систем. Відсутність необхідності фізичного контакту також сприяє підвищенню комфорту під час використання.

Зручність користувача під час процесу автентифікації є важливим критерієм, який не можна ігнорувати при виборі методу ідентифікації для системи безпеки. Оскільки користувачі взаємодіятимуть із цією системою кілька разів на день, щодня, важливо забезпечити їм як фізичний комфорт під час використання, так і психологічну зручність. Автентифікація, яка викликає дискомфорт або є незручною у виконанні, може негативно вплинути на сприйняття системи та її практичність у повсякденному використанні.

Ще одним аспектом є економічна складова системи. Вартість реалізації залежить від функціонального призначення системи, складності обраних методів ідентифікації, використання додаткового обладнання, спрямованого на підвищення рівня безпеки, та інших факторів. Успішний вибір методу передбачає врахування балансу між рівнем безпеки, комфортом користувачів та економічною ефективністю.

Споживання ресурсів пристрою, зокрема батареї, є важливим критерієм при виборі біометричного методу ідентифікації, особливо в мобільних системах. Складність обробки даних, що використовуються для ідентифікації, впливає на енергоспоживання пристрою. Наприклад, методи, які вимагають обробки великих обсягів даних або виконання складних математичних операцій, можуть значно збільшувати витрати заряду батареї. Особливо це стосується систем, які постійно працюють у фоновому режимі для моніторингу або забезпечення швидкого доступу до функцій автентифікації. Мінімізація використання енергоємних алгоритмів та оптимізація роботи сенсорів може суттєво знизити навантаження на акумулятор, забезпечуючи більш тривалу автономну роботу пристрою.

Всі названі нами критерії зобразимо у вигляді таблиці. Порівнявши дані кожного з методів зможемо дійти висновку, який критерій краще обрати [14]:

Таблиця 2.1

Порівняльна характеристика сучасних методів біометричної автентифікації

| Метод | ППР,% | ПДР,% | Можливість фальсифікації | Стійкість характеристик | Стійкість до середовища | Швидкість ідентифікації | Безконтактна ідентифікація | Комфорт | Вартість | Енергоефективність |
|---|---|---|---|---|---|---|---|---|---|---|
| Відбиток пальця | 0.5 | 0.0004 | С | С | Н | В | Н | С | С | В |
| Геометрія обличчя | 2.7 | 0.3 | В | С | Н | С | В | В | Н | С |
| Термограма обличчя | 4.1 | 0.5 | С | С | Н | С | В | С | В | Н |
| Райдужна оболонка | 0.023 | 0.00002 | Н | В | С | В | В | В | В | С |
| Сітківка ока | 0.012 | 0.0001 | Н | В | С | Н | Н | Н | В | Н |

У табл. 2.1 скорочення «Н» означає «Низький», «С» означає «Середній», «В» означає «Високий».

Як бачимо за результатами порівняння, метод ідентифікації особи за відбитком пальця має посередні результати в випробуванні на помилки першого та другого роду. Окрім цього, з використанням сучасних технологій стає

можливим створення підробок відбитку та використання їх для подолання системи безпеки. До того ж, пил, забруднення, теплові коливання а також надмірна вологість можуть впливати на роботу сканера та перешкоджати успішній авторизації. Також до недоліків можна віднести обов'язковий фізичний контакт зі сканером, що має негативний вплив на гігієнічність та користувацький досвід в цілому. Проте, даний метод пропонує перелік переваг, що позитивно виділяють його посеред альтернатив. Сканери, необхідні для зчитування відбитків наразі присутні у будь-якому сучасному смартфоні чи планшеті, що повністю прибирає необхідність задумуватися про додаткове обладнання при використанні цього методу на мобільних платформах. Ідентифікація проходить практично миттєво, що сильно сприяє комфорту користувачів. Окрім цього, ідентифікація за відбитком пальця виділяється як найбільш енергоефективний метод з запропонованих, за рахунок швидкої авторизації та мінімального використання електроенергії, що робить їх особливо вигідними для використання на мобільних платформах. Хоча відбиток пальця і схильний до змін внаслідок старіння або фізичного втручання та травм, базові характеристики все одно залишаються стабільними протягом багатьох років, що робить їх підходящими для довгострокового використання.

Отже, в результаті розгляду сучасних методів біометричної автентифікації, можемо дійти висновку, що метод ідентифікації за відбитком пальця є оптимальним вибором для використання в застосунку менеджері паролів на мобільних платформах. За умови відсутності необхідності в безконтактній ідентифікації, решта недоліків меркне на фоні швидкості, комфорту та ефективності використання даного методу.

### 2.5 Аналіз технологій ідентифікації за відбитком пальця

У кожному відбитку пальця людини можна визначити два типи ознак – глобальні й локальні.

Глобальні ознаки (рис.2.1) – ті, які людина може побачити неозброєним оком:

- Папілярний візерунок. Область візерунка – виділений фрагмент відбитка, в якому локалізовані всі глобальні ознаки.
- Ядро або центр – точка, локалізована в середині відбитка або певної виділеної області.
- Пункт «дельта» – початкова точка. Це місце, в якому відбувається поділ або поєднання борозенок папілярних ліній, чи дуже коротка борозенка (може доходити до крапки).
- Тип лінії – дві найбільші лінії, що починаються як паралельні, а потім розходяться і огинають всю область образу.
- Лічильник ліній – це число ліній на області образу, чи між ядром і пунктом «дельта».

Є наступні типи папілярних візерунків:

- візерунки типу «петля» (права, ліва, подвійна, центральна);
- візерунки типу «дельта» чи «дуга» (проста і гостра);
- візерунки типу «спіраль» (центральна і змішана).

Інший тип ознак – локальні ознаки. Їх називають мінуції (рис.2.2) (особливостями або особливими точками) – унікальні для кожного відбитку пальця ознаки, які визначають пункти зміни структури папілярних ліній (роздвоєння, закінчення, розрив тощо), орієнтацію папілярних ліній і координати в цих пунктах. Кожен відбиток може містити до 70 і більше мінуцій.


Рис. 2.1. Глобальні ознаки відбитку пальця


Рис. 2.2. Локальні ознаки відбитку пальця(мінуції)

Практика показує, що відбитки пальців у різних людей можуть мати окремі однакові глобальні ознаки, проте абсолютно неможливо наявність однакових мікровізерунків мінуцій. Тому глобальні ознаки використовують з метою поділу бази даних на класи і на етапі автентифікації. На другому етапі розпізнавання користуються вже локальними ознаками.

На даний час в сфері дактилоскопії в основному використовуються стандарти ANSI і ФБР США. У них визначені такі вимоги до образу відбитка:

- кожен образ представлено у форматі нестислого TIFF;
- образ повинен мати дозвіл не менше 500 dpi;
- образ має бути напівтоновим з 256 рівнями яскравості;
- максимальний кут повороту для відбитка від вертикалі – не більше 15 град.;
- основні типи мінуцій – закінчення і роздвоєння.

На даний час існує декілька класів алгоритмів порівняння відбитків пальців, основні з яких розглянемо далі.

### 2.5.1 Кореляційне порівняння

Суть даного методу полягає в тому, що отриманий зі сканера відбиток пальця накладається на кожен стандарт з бази даних по черзі, після чого прямо по пікселях зображень здійснюється прорахунок відмінностей між ними.

Щоправда, при цьому доводиться враховувати один момент. Справа в тому, що людина щоразу прикладає палець під різними кутами і не точно в те саме місце робочої області сканера. А це означає, що процес порівняння відбитка його пальця з еталонами повинен включати безліч ітерацій, на кожній з яких зображення, отримане зі сканера, повертається під невеликим кутом або трохи зміщується. Таким чином, підраховується кореляція (за рівнем інтенсивності) між відповідними пікселями, обчислена для різних вирівнювань зображень відносно один одного (наприклад, шляхом різних зміщень та обертань); за відповідним коефіцієнтом приймається рішення про ідентичність відбитків.

Нехай T і I два зображення відбитка пальця, що відповідають шаблонному та введеному зображення відповідно. Тоді їхня різниця – це сума квадрата різниці (SSD) між значеннями яскравості відповідних пікселів [10]:

$$SSD(T,I) = ||T-I||^2 = (T-I)^T * (T-I) = \left|\left|T\right|\right|^2 + \left|\left|I\right|\right|^2 - 2 * T^T * I, \qquad (2.40)$$

де надрядковий індекс T означає транспонування вектора.

Якщо терми $\left|\left|T\right|\right|^2$ та $\left|\left|I\right|\right|^2$ є константами, то різниця між двома зображеннями мінімізується, коли взаємна кореляція (CC) максимізується:

$$CC(T,I) = T^T * I, \qquad (2.41)$$

CC(T,I) постає як третій терм у рівнянні 2.39. Взаємна кореляція (проста кореляція) – міра подібності зображень. Внаслідок переміщення та ротації, які неминуче характеризують два різних зображення відбитка з одного пальця, їх схожість не може бути легко обчислена шляхом накладання зображень T та I один на одного та застосуванням рівняння 2.40.

Нехай $I\ (\Delta_x, \Delta_y, \theta)$ представляє ротацію вхідного зображення I під кутом θ навколо джерела, у ролі якого виступає центр зображення, і зсувається (переміщається) на $\Delta_x, \Delta_y$ в напрямку x та y відповідно. У цьому випадку

подібність між двома зображеннями відбитків пальців I та T може бути розрахована як:

$$S(T,I) = \max \Delta_x, \Delta_y, \theta\, CC(T, I(\Delta_x, \Delta_y, \theta)), \qquad (2.42)$$

Безпосереднє застосування рівняння 2.41 рідко призводить до прийнятних результатів з наведених нижче причин:

- Нелінійне спотворення робить відбиток з одного й того ж пальця значно різним у термах глобальної структури.
- Стан шкіри та ступінь тиску натискання пальця впливають на яскравість зображення, контрастність і товщину виступів, значно видозмінюючи різні відбитки.
- Пряме використання рівняння 2.41 має велику обчислювальну складність.

З метою спрощення обчислювальної складності кореляційної техніки застосовуються інтелектуальні підходи задля досягнення ефективної реалізації.

Кореляційна теорема [15] свідчить, що обчислення кореляції у просторовій області (оператор $\bigoplus$) еквівалент виконання крапкового множення у розподілі Фур'є, зокрема:

$$T \bigoplus I = F^{-1}\big(F^*(T) * F(I)\big), \qquad (2.43)$$

де F – перетворення Фур'є вищезгаданого зображення;

Результатом рівняння 2.42 є кореляційне зображення, значення якого в пікселі [x,y] відповідає кореляції між T та I, коли зміщення $\Delta_x =$ x і $\Delta_y =$ y.

### 2.5.2 Зіставлення за шаблоном

Алгоритм зіставлення за шаблоном є поширеним підходом для ідентифікації відбитків пальців, де враховуються не лише ключові точки (мінюції), але й глобальні характеристики папілярного візерунка. Це дозволяє

підвищити точність і стійкість до часткових пошкоджень або спотворень відбитків [16]. У цьому методі використовується порівняння глобальних та локальних характеристик, таких як:

- Щільність гребенів і борозенок.
- Кривизна папілярних ліній.
- Товщина ліній.
- Орієнтаційне поле відбитка.

На основі цих характеристик створюється матриця ознак, яка відображає унікальність кожного відбитка.

Після зчитування зображення відбитка I(x,y) та проведення нормалізації за формулою 2.1, проводиться обчислення орієнтаційного поля, що визначає напрямок гребенів в кожній точці зображення.

Для градієнтів зображення обчислюються часткові похідні:

$$G_x = \frac{\partial I`}{\partial x},\ G_y = \frac{\partial I`}{\partial y}, \tag{2.44}$$

Зображення ділиться на блоки $B_{i,j}$, для кожного з яких обчислюється напрямок:

$$\theta_{i,j} = \frac{1}{2}\arctan\left(\frac{2\sum G_x G_y}{\sum G_x^2 - \sum G_y^2}\right), \tag{2.45}$$

Задля виділення текстурних характеристик щільність ліній визначається як середня відстань між гребенями в локальному блоці:

$$\rho_{i,j} = \frac{1}{n}\sum_{k=1}^{n} d_k\ , \tag{2.46}$$

де $d_k$ – відстань між сусідніми гребенями.

Кривизна оцінюється через похідну зміни кута орієнтації:

$$k_{i,j} = \frac{\partial^2 \theta_{i,j}}{\partial x^2} + \frac{\partial^2 \theta_{i,j}}{\partial y^2}, \quad (2.47)$$

Для визначення товщини ліній обчислюється ширина гребеня за результатами бінаризації та морфологічних операцій:

$$\omega_{i,j} = width\ of\ the\ ridge\ at\ block\ B_{i,j}, \quad (2.48)$$

Формування глобального шаблону – складаємо матрицю ознак:

$$T = \{(\theta_{i,j}, \rho_{i,j}, k_{i,j}, \omega_{i,j}) | i, j = 1,2, \dots, N\}, \quad (2.49)$$

де N – кількість блоків зображення.

Далі відбувається вирівнювання шаблонів. Враховується можливе зміщення, обертання або масштабування між шаблонами. Для цього використовується метод крос-кореляції:

$$C(\Delta x, \Delta y) = \sum_{i,j} T_1\,(i,j) * T_2(i + \Delta x, j + \Delta y), \quad (2.50)$$

де $T_1$ і $T_2$ – матриці шаблонів для двох відбитків.

Загальний збіг визначається через функцію схожості:

$$S = \frac{\sum_{i,j} T_1(i,j) *\ T_2(i,j)}{\left||T_1|\right| * ||T_2||}, \quad (2.51)$$

де ‖T‖ – норма матриці шаблона.

Якщо $S \geq T_{match}$, то вважається, що відбитки збігаються. $T_{match}$ – порогове значення, що визначає чутливість системи.

2.5.3 Порівняння на основі графів

Алгоритм зіставлення на основі графів є одним із сучасних підходів, що використовує топологічну структуру папілярного візерунка. У цьому методі мінюції та їх взаємне розташування представляються у вигляді графа, де вершини – це мінюції, а ребра – зв'язки між ними. Це забезпечує стійкість до спотворень і можливість порівняння навіть часткових відбитків.

Відбиток пальця представляється як неорієнтований граф [17] G = (V,E), де: $V = \{v_1, v_2, \dots, v_n\}$ – множина вершин, які відповідають мінюціям.

$E = \{(v_i, v_j) | i, j = 1, \dots, n\}$ – множина ребер, які описують зв'язки між мінюціями.

Кожна вершина $v_i$ характеризується координатами та властивостями мінюції:

$$v_i = (x_i, y_i, \theta_i, t_i), \tag{2.52}$$

де $x_i, y_i$ – координати мінюцій;

$\theta_i$ – орієнтація мінюції;

$t_i$ – тип мінюції (розгалуження, закінчення тощо).

Кожне ребро $e_{ij}$ описує відстань і кут між двома мінюціями:

$$e_{ij} = \left(\Delta_x, \Delta_y, \Delta_\theta\right), \tag{2.53}$$

де $\Delta_x = x_j - x_i$;

$\Delta_y = y_j - y_i$;

$\Delta_\theta = \theta_j - \theta_i$.

Для опису графу використовується матриця суміжності A, елементи якої визначають зв'язок між вершинами:

$$A_{ij} = \begin{cases} 1, & \text{якщо} \left(v_i, v_j\right) \in E, \\ 0, & \text{інакше.} \end{cases}, \tag{2.54}$$

Алгоритм починається з визначення мінюцій за допомогою традиційних методів: скелетизації зображення, бінаризації та аналізу хрестового числа (CN).

Кожна мінюція $M_i$ перетворюється на вершину $v_i$ з відповідними характеристиками.

Ребра встановлюються між мінюціями, які перебувають у межах заданого радіуса R:

$$R_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \text{якщо } R_{ij} \leq R, \quad (2.55)$$

Головна задача – знайти відповідність між графом-взірцем $G_1 = (V_1, E_1)$ і графом-тестом $G_2 = (V_2, E_2)$. Для цього використовують наступні методи:

1. Пошук ізоморфізму субграфів:

Алгоритм шукає максимальний загальний підграф G` = (V`, E`), де: $G` \subseteq G_1$ і $G` \subseteq G_2$.

Це завдання часто за допомогою пошуку ізоморфізму:

$$\phi: V_1 \to V_2 \quad \text{та} \quad \phi(E_1) = E_2, \quad (2.56)$$

2. Оцінка відповідності ребер:

Відповідність між двома вершинами $v_i \in V_1$ і $v_j \in V_2$ вважається успішною, якщо виконуються умови:

$$\left|\left|x_i - x_j\right|\right| \leq \epsilon_x, \quad \left|\left|y_i - y_j\right|\right| \leq \epsilon_y, \quad ||\theta_i - \theta_j|| \leq \epsilon_\theta, \quad (2.57)$$

де $\epsilon_x, \epsilon_y, \epsilon_\theta$ – допустимі пороги.

3. Функція подібності:

Порівняння двох графів виконується через функцію подібності:

$$S = \frac{|V`|}{\max(|V_1|, |V_2|)}, \quad (2.58)$$

де |V`| – кількість спільних вершин;

$|V_1|, |V_2|$ – кількість вершин у графах.

Результат зіставлення визначається на основі порога $T_{match}$:

$$Match, \text{якщо } S \geq T_{match}. \quad (2.59)$$

## 2.6 Порівняльний аналіз рішень з управління паролями

Оскільки виходячи з попередньо проаналізованої інформації ми впевнилися в необхідності використання менеджеру паролів для сучасної людини, а також визначились з технологіями автентифікації, що ми будемо використовувати, перед нами постала задача з встановлення потреб, що на даний момент не є закритими. З цією метою розглянемо декілька найпопулярніших програмних рішень з управління паролями:

LastPass – це один із найвідоміших і найстаріших менеджерів паролів на ринку. Він пропонує як безкоштовну, так і платну версії. Безкоштовна версія дозволяє зберігати необмежену кількість паролів і синхронізувати їх між пристроями, але має певні обмеження на кількість пристроїв, які можна підключити одночасно. Платна версія додає додаткові функції, такі як аварійний доступ, сімейний обмін паролями та безпечне зберігання нотаток. LastPass відзначається простотою використання та широкою сумісністю з різними платформами і браузерами.

1Password позиціонується як один із найбезпечніших менеджерів паролів. Він використовує сильне шифрування і пропонує додаткові функції безпеки, такі як двофакторна аутентифікація та біометрична розблокування. Крім зберігання паролів, 1Password дозволяє зберігати кредитні картки, нотатки та інші дані. Інтерфейс програми дуже зручний і приємний для користувача. Однак, 1Password є одним із найдорожчих менеджерів паролів на ринку.

Bitwarden – це відкритий менеджер паролів, що означає, що його код доступний для загального огляду. Це підвищує довіру до безпеки програми, оскільки будь-хто може перевірити її код на наявність вразливостей. Bitwarden пропонує як безкоштовну, так і платну версії. Безкоштовна версія має широкий

функціонал і дозволяє синхронізувати паролі між різними пристроями. Платна версія додає додаткові функції, такі як двофакторна аутентифікація і підтримка WebAuthn.

Dashlane – це комплексне рішення для захисту даних в Інтернеті. Крім менеджера паролів, він пропонує VPN, моніторинг темного веб і інші інструменти для захисту даних. Dashlane має інтуїтивний інтерфейс і автоматично оновлює слабкі паролі. Програма також пропонує функцію безпечного спільного використання паролів.

Keeper фокусується на безпеці. Він використовує додаткові рівні захисту, такі як зашифровані сховища і біометрична аутентифікація. Keeper також пропонує корпоративні рішення для бізнесу. Крім зберігання паролів, Keeper дозволяє зберігати файли, нотатки і навіть здійснювати безпечне спілкування.

Задля облегшення порівняння оберемо критерії, на основі яких будемо проводити порівняння:

- Функціональність – включає набір функцій, таких як зберігання паролів, автоматичне заповнення форм, синхронізація між пристроями, підтримка біометричної автентифікації, генерація паролів, збереження файлів та інтеграція з іншими додатками.
- Ціна – вартість підписки, безкоштовні функції, наявність безкоштовного плану або пробного періоду.
- Інтерфейс – зручність користування, дизайн, доступність мов, простота налаштування.
- Рівень захисту при втраті пристрою – розглядається, чи підтримується двофакторна та біометрична автентифікація, можливість дистанційного очищення даних, та інші заходи, які знижують ризик компрометації облікового запису.
- Відкритість коду – забезпечує прозорість роботи програми, дозволяє спільноті розробників шукати й усувати вразливості, а також надає можливість адаптувати програму для власних потреб.

За обраними нами критеріями, створимо та розглянемо порівняльну таблицю 2.2:

Таблиця 2.2

Порівняльна характеристика сучасних рішень з менеджменту паролів

| Менеджер паролів | LastPass | 1Password | Bitwarden | Dashlane | Keeper |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| Функціональність | Зберігання паролів, автоматичне заповнення, генерація паролів, синхронізація (у платному плані). | Сильна підтримка сімейних і бізнес-акаунтів, генерація паролів, збереження документів, поділ на "сховища". | Синхронізація на всіх пристроях, генерація паролів, збереження нотаток. | Зберігання паролів, VPN у преміум-плані, автоматична зміна паролів, перевірка слабких паролів | Фокус на безпеку, підтримка збереження файлів, історія записів, приватні повідомлення. |
| Ціна | Безкоштовний план, Premium $3/місяць. | Безкоштовний план, Premium $10/рік. | Безкоштовний план, Premium $10/рік. | Безкоштовний план, Premium $10/рік. | Від $2.91/місяць, окремо захист файлів. |
| Інтерфейс | Інтуїтивний, але були скарги на перевантаженість інтерфейсу. | Чистий, зручний, підтримує різні платформи. | Простіший інтерфейс, менш сучасний вигляд.. | Сучасний, але іноді складний у налаштуванні. | Сучасний, фокус на безпеку, доступність інтеграції з іншим ПЗ. |

Закінчення таблиці 2.2

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| Рівень захисту при втраті пристрою | 2FA, можливість відновлення облікового запису через головний пароль. | 2FA, можливість встановлення параметра "Автоблокування" | Підтримує 2FA | 2FA, повідомлення про компрометацію даних. | Підтримка 2FA, віддалене очищення даних, аудит безпеки |
| Відкритість коду | Закритий код | Закритий код | Відкритий код | Закритий код | Закритий код |
| Історія вразливостей | Є історія витоків (2022). | Були випадки порушення безпеки (2023). | Без історії серйозних витоків | Відомі незначні вразливості. | Жодних відомих витоків. |
| Алгоритм шифрування | AES-256 | AES-256 | AES-256 | AES-256 | AES-256 |

Як бачимо з табл. 2.2, розглянуті нами застосунки є впевненими лідерами в областях дизайну, інтерфейсу та цілком зручності для користувача, а також загального функціоналу. І хоча їх цінова політика може відлякнути деяких користувачів, основним аспектом для менеджеру паролів залишається захист інформації. Розглянуті нами застосунки мають високий рівень захисту від шкідливого програмного забезпечення, завдяки сильним алгоритмам шифрування, проте їх захист в разі втрати чи викрадення пристрою залишає бажати кращого. З передбачених механізмів захисту всі вони підтримують 2FA, проте, нажаль, це не є панацеєю. До того ж, алгоритм шифрування AES-256, котрим користуються всі розглянуті застосунки, має значну вразливість в разі отримання зловмисниками доступу до ключа шифрування. Окрім цього, всі проаналізовані застосунки зберігають дані користувача в хмарних сховищах. І хоча це надає перевагу у вигляді кросплатформеності, загальний рівень захисту

даних користувача значно знижується, що підтверджується відомими випадками витоку інформації, що зберігалася в хмарних сховищах. З усіх застосунків лише Bitwarden має відкритий код, що робить його привабливим для користувачів, які цінують прозорість і бажають мати повний контроль над захистом своїх даних.

Загалом, на мою думку, саме рухаючись в напрямку аспектів захисту, не закритих популярними рішеннями на ринку, можна виявити нереалізований потенціал в сфері менеджерів паролів та досягти нового рівня гарантій безпеки інформації користувачів. Саме з цією метою створимо менеджер паролів, що гарантує безпеку даних користувачів завдяки сучасному методу автентифікації за біометрією, а також впровадженню додаткового рівня шифрування даних та зберіганню даних в захищеному локальному сховищі.

### 2.7 Висновки до другого розділу

В цьому розділі було проаналізовано методи біометричної автентифікації. З розглянутих варіантів метод ідентифікації за відбитком пальця було обрано як оптимальний для застосування в застосунку менеджері паролів.

Окрім цього, було описано найбільш вживані алгоритми порівняння відбитків пальців. Метод порівняння за мінюціями є найефективнішим з огляду на низку ключових переваг над іншими методами.

Порівняння за мінюціями базується на обмеженій кількості унікальних точок (зазвичай кілька десятків), що дозволяє зменшити обсяг обчислень і прискорити процес. Натомість кореляційне порівняння вимагає аналізу всього зображення, включаючи текстуру, що збільшує обчислювальну складність. Алгоритм мінюцій адаптується до трансляції, обертання та масштабування зображення завдяки нормалізації координат мінюцій. Кореляційні методи менш стійкі до таких спотворень, оскільки вони чутливі до змін у відображенні або умовах сканування. Метод мінюцій може працювати навіть із частково пошкодженими відбитками, оскільки йому потрібні лише локалізовані точки

розгалужень і закінчень. Кореляційне порівняння потребує високоякісних зображень без значних спотворень.

Мінюції є унікальними для кожного відбитка і забезпечують високу точність ідентифікації. Зіставлення за шаблоном базується на загальних особливостях (наприклад, текстурі або основних візерунках гребенів), що знижує його точність у випадках, коли відбитки частково накладаються чи мають спотворення. Мінюції дозволяють скоротити обсяг інформації, що використовується для порівняння, до набору координат та орієнтацій точок. Шаблонне зіставлення вимагає збереження значно більших обсягів даних, що неефективно для мобільних застосунків.

Порівняння мінюцій займає менше часу ніж порівняння за графами, оскільки воно зосереджене на ключових точках, тоді як метод графів аналізує відношення між усіма гребенями та мінюціями, що значно ускладнює процес. Як наслідок, метод мінюцій більш ефективний для застосунків на мобільних платформах, таких як Android, оскільки вимагає менше пам’яті та обчислювальної потужності.

Окрім цього, було проаналізовано найпопулярніші рішення на ринку менеджерів паролів та визначено їх ключовий недолік в вигляді недостатнього рівня захисту пристрою в разі його втрати чи викрадення.

# РОЗДІЛ 3
# ТЕХНІЧНА РЕАЛІЗАЦІЯ ТА АЛГОРИТМИ РОБОТИ ЗАСТОСУНКУ

## 3.1 Аналіз використаних технологій

Для реалізації застосунку менеджеру паролів було застосовано наступні технології мови програмування Kotlin:

- Room Persistence Library
- FusedLocationProviderClient
- BiometricPrompt Api
- CP-ABE
- Java Pairing Based Cryptography

Далі розглянемо ці технології детальніше.

### 3.1.1 Room Persistence Library

Room Persistence Library – це бібліотека в пакеті Android Jetpack, за допомогою якої реалізується використання технології Об'єктно-реляційного відображення (ORM), яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних». Room скорочує обсяг типового коду, який необхідно написати для інтеграції з SQLite, та автоматизує багато завдань, таких як обробка запитів та керування транзакціями [20].

Основні компоненти Room:

- Entity – це клас, що представляє таблицю у базі даних. Кожному об'єкту класу відповідає 1 рядок у таблиці.
- DAO (об'єкт доступу до даних) – це інтерфейс або абстрактний клас, який визначає запит до бази даних. Room автоматично генерує їх реалізацію.
- База даних – це абстрактний клас, що поєднує entity та dao. Це точка входу для доступу до бази даних.

Room також підтримує перевірку SQL запитів під час компіляції, що дозволяє уникнути помилок ще до виконання програми. Однією з переваг бібліотеки є автоматичне керування потоками – Room забороняє запити до бази даних в основному потоці, щоб уникнути блокування UI [21].

У створеному застосунку Room використовується для зберігання та обробки даних користувачів та записів паролів. Вона є основою для роботи з базами даних і забезпечує простий, інтуїтивно зрозумілий і ефективний спосіб взаємодії з локальними даними. Room автоматично керує багатьма аспектами, такими як створення таблиць, оновлення та перевірка запитів, що значно полегшує розробку.

Структура бази даних базується на Entity. У нашій програмі це 2 важливих об'єкта: User (користувач) і PasswordEntry (запис про пароль). Вони відповідають таблицям у базі даних, і кожен об'єкт класу представляє окремий запис (рядок). Наприклад, у таблиці користувачів зберігаються такі атрибути, як ім'я користувача, пароль та статус адміністратора, а в таблиці паролів зберігаються відомості про сайт, логін, пароль та прив’язку до конкретного користувача.

Для взаємодії з базою даних використовується DAO (Data Access Object). DAO-це інтерфейс, який записує методи для виконання таких операцій, як додавання нових записів до таблиць, їх видалення або отримання даних на основі певних умов. Room автоматично генерує код SQL на основі визначень DAO, зменшуючи ймовірність помилок та полегшуючи обслуговування коду.

База даних управляється через основний клас RoomDatabase, який об'єднує Entity та Dao. У нашому застосунку цей клас також забезпечує єдиний вхідний пункт для доступу до бази, використовуючи патерн *singleton*. Це запобігає створенню декількох екземплярів бази даних і гарантує, що всі операції зберігання виконуються через один і той самий об'єкт.

Для забезпечення асинхронної роботи було інтегровано Room з LiveData та корутинами. LiveData забезпечує автоматичне оновлення інтерфейсу користувача, коли змінюються дані в базі. Наприклад, коли користувач додає новий запис про пароль, ця зміна одразу відображається у списку без

необхідності вручну оновлювати UI. Корутини дозволяють виконувати всі операції з базою даних у фоновому потоці, що запобігає блокуванню головного потоку та забезпечує плавність роботи застосунку.

У створеному застосунку Room виконує роль зручного та гнучкого інструмента для зберігання даних. Він дозволяє легко реалізувати такі функції, як автентифікація користувачів, доступ до записів паролів і шифрування даних перед їхнім збереженням. Завдяки автоматизованій перевірці запитів і управлінню потоками, Room значно спрощує розробку та підтримку програми.

### 3.1.2 FusedLocationProviderClient

FusedLocationProviderClient – це API в рамках Google Play Services, який забезпечує доступ до геолокації пристрою. Він реалізує технологію «злиття», котра використовує алгоритми машинного навчання та сигнальну обробку, щоб оцінювати й комбінувати інформацію з різних джерел. Ця технологія поєднує такі джерела, як GPS, Wi-Fi, мобільні мережі та сенсори пристрою щоб зменшити споживання енергії та збільшити точність, що робить її ідеальним вибором для додатків, які потребують геолокаційних послуг.

Однією з головних переваг FusedLocationProviderClient є його гнучкість. Розробники можуть налаштувати частоту запитів, бажану точність (наприклад, "висока точність" або "тривале відстеження з низьким енергоспоживанням") та час очікування [21]. Завдяки такій конфігурованості API підходить як для застосунків реального часу (наприклад, карти), так і для менш вимогливих сценаріїв, таких як аналіз переміщень.

Ще одним важливим аспектом є його асинхронна архітектура. Клієнт використовує *callbacks* та *listeners*, щоб відправляти дані про місцезнаходження у фоновому режимі, не блокуючи основний потік програми [20]. Це дозволяє інтегрувати геолокаційні функції навіть у складні програми, забезпечуючи при цьому високу продуктивність.

FusedLocationProviderClient також підтримує сучасні стандарти безпеки. Наприклад, для доступу до геолокаційних даних користувач повинен надати необхідні дозволи (таких як *ACCESS_FINE_LOCATION*). Цей процес доповнюється динамічним запитом дозволів у реальному часі, що гарантує конфіденційність даних користувачів.

У нашому застосунку FusedLocationProviderClient використовується для забезпечення перевірки геолокації як одного з атрибутів доступу до системи. Ця функція реалізована для контролю відповідності місцезнаходження користувача до попередньо зафіксованої геолокації під час першої авторизації. Якщо різниця в координатах перевищує певний поріг, доступ блокується.

Ключова інтеграція API включає такі етапи:

1. Запит поточної геолокації**.** Під час першої авторизації ми ініціюємо FusedLocationProviderClient для отримання поточних координат користувача. Це здійснюється через функцію getCurrentLocation, яка надає найбільш актуальне місцезнаходження з використанням усіх доступних джерел (GPS, Wi-Fi тощо).
2. Обчислення відстані. Під час кожної наступної авторизації отримані координати порівнюються з початковими. Це порівняння виконується за допомогою Haversine Formula, яка визначає відстань між двома точками на сфері. Формула виглядає так:

$$d = 2r * \arcsin\left(\sqrt{sin^2\left(\frac{\Delta\varphi}{2}\right) + cos(\varphi_1) * \cos(\varphi_2) * sin^2\left(\frac{\Delta\lambda}{2}\right)}\right), \qquad (3.1)$$

де r – радіус Землі (приблизно 6371 км);

$\varphi_1, \varphi_2$ – широти;

$\Delta\varphi$ та $\Delta\lambda$ – різниці в широтах і довготах відповідно.

3. Перевірка та рішення. Якщо обчислена відстань перевищує допустимий поріг (в нашому випадку, 1км), авторизація блокується. Це забезпечує захист системи від несанкціонованого доступу з віддалених місць.

4. Асинхронність роботи. Завдяки використанню корутин, запит на геолокацію виконується у фоновому режимі, щоб уникнути блокування основного потоку. Цей підхід гарантує плавність роботи програми, навіть якщо отримання координат займає кілька секунд.

Ключовою перевагою інтеграції FusedLocationProviderClient у нашому застосунку є підвищена безпека. Використання геолокації як атрибуту доступу ускладнює зловмисникам можливість обійти автентифікацію. Крім того, такий підхід демонструє можливості сучасних API для створення адаптивних і безпечних програм.

### 3.1.3 BiometricPrompt API

BiometricPrompt API – це інструмент в Android, який реалізує різноманітні технології біометричної автентифікації, такі як розпізнавання геометрії обличчя, райдужної оболонки ока, та, в нашому випадку – відбитків пальців, і забезпечує їх інтеграцію в мобільні додатки. BiometricPrompt надає стандартизований інтерфейс, який дозволяє взаємодіяти з апаратним забезпеченням пристрою, що відповідає за біометричну перевірку, та знижує складність інтеграції таких функцій у додаток.

Однією з головних переваг BiometricPrompt є високий рівень безпеки. API підтримує механізми автентифікації на базі апаратного забезпечення, таких як Trusted Execution Environment (TEE) або Secure Element (SE). Це гарантує, що біометричні дані користувача ніколи не залишають захищену область пристрою, навіть якщо зловмисник отримав доступ до пам’яті додатка. Дані автентифікації ніколи не передаються через мережу або на зовнішні сервери [21].

Іншим важливим аспектом є гнучкість і зручність використання BiometricPrompt. API автоматично створює діалогове вікно, яке відповідає стилю та політикам безпеки Android. Це означає, що розробники можуть уникнути складного кодування інтерфейсу для взаємодії з користувачем, фокусуючись на

функціональності програми. Крім того, API підтримує зворотну сумісність зі старими версіями Android через Jetpack-бібліотеки.

BiometricPrompt також дозволяє налаштувати процес автентифікації, задаючи рівень довіри. Наприклад, розробник може вимагати лише strong біометрію (висока надійність, як у відбитків пальців), або дозволяти слабші механізми (наприклад, розпізнавання обличчя без додаткового апаратного забезпечення), в залежності від цілей програми.

У створеному додатку BiometricPrompt API використовується для автентифікації адміністратора, забезпечуючи підвищений рівень безпеки для доступу до даних. Цей API дозволяє адміністратору швидко входити в систему за допомогою відбитка пальця, оминаючи введення логіна та пароля.

Під час запуску програми BiometricPrompt ініціює перевірку. Якщо користувач є адміністратором, викликається метод автентифікації через відбиток пальця. Біометричні дані користувача не зберігаються в програмі чи базі даних; вони залишаються в захищеній зоні пристрою. Це забезпечує відповідність вимогам конфіденційності.

BiometricPrompt викликає біометричний діалог через функцію authenticate. Коли користувач успішно проходить автентифікацію, програма отримує callback із результатом. Якщо автентифікація не вдається (наприклад, користувач відхиляє запит або біометричні дані не співпадають), програма блокує доступ і пропонує альтернативний метод входу через логін і пароль.

BiometricPrompt в нашій програмі налаштований для використання лише сильних біометричних даних (strong biometrics). Це дозволяє зменшити ризик використання менш захищених методів, наприклад, програмного розпізнавання обличчя.

Біометрична автентифікація також поєднується з атрибутивним шифруванням у нашій програмі. Після успішної автентифікації програма перевіряє атрибути доступу (наприклад, isAdmin) та геолокацію, перш ніж надати доступ до зашифрованих даних. Це дозволяє створити комплексну систему безпеки, де біометричний доступ є лише однією зі складових.

BiometricPrompt API у нашій програмі забезпечує швидку, зручну та високозахищену автентифікацію для адміністратора. Завдяки використанню цього API можна гарантувати, що доступ до програми можливий лише для авторизованих користувачів із довіреними біометричними даними. Це відповідає сучасним стандартам безпеки та забезпечує комфорт для кінцевих користувачів.

### 3.1.4 CP-ABE

Для посилення безпеки інформації в створюваному застосунку використаємо метод Атрибутивного шифрування (ABE). ABE – це криптографічний метод, в якому доступ до зашифрованих даних визначається на основі набору атрибутів. Такий підхід дозволяє гнучко керувати правами доступу, що особливо корисно в системах з великою кількістю користувачів та різнорівневим доступом до інформації [18].

ABE забезпечує шифрування, де дані можна розшифрувати лише тоді, коли набір атрибутів користувача відповідає політиці доступу, зашифрованій у даних. Розрізняють два основних підвиди ABE:

1. Шифрування на основі політик (Ciphertext-Policy ABE): політика доступу вбудовується у шифротекст, а атрибути прив'язані до ключа користувача.
2. Шифрування на основі атрибутів ключа (Key-Policy ABE): політика доступу прив'язується до ключа, а атрибути до шифротексту.

Основний алгоритм дії ABE:

1. Setup (ініціалізація): генерується система публічних і приватних параметрів.
   - Вхід: параметр безпеки λ.
   - Вихід: параметри публічного ключа PK та секретного ключа авторитету MK.
2. KeyGen (генерація ключа): створює приватний ключ для користувача на основі набору атрибутів.

- Вхід: MK (секретний ключ авторитету), S (набір атрибутів користувача).
- Вихід: приватний ключ користувача $SK_s$.

3. Encrypt (шифрування): зашифровує дані згідно з політикою доступу (CP-ABE) або атрибутами (KP-ABE).
   - Вхід: Повідомлення M, політика доступу T (або атрибути).
   - Вихід: Шифротекст CT
4. Decrypt (розшифрування): розшифровує дані, якщо набір атрибутів S відповідає політиці T.
   - Вхід: CT (шифротекст), $SK_s$ (ключ користувача).
   - Вихід: Повідомлення M або повідомлення про відмову.

Для нашого застосунку оберемо саме CP-ABE, оскільки цей метод синергічно співпрацюватиме з біометричною автентифікацією, а також дозволить інтегрувати додатковий рівень захисту з перевіркою геолокації.

Математичний опис CP-ABE [19]:

1. Ініціалізація: генеруються групи $G_1, G_2$ порядку p з білінійним відображенням e: $G_1 \times G_1 \to G_2$. Вибираються генератор $g \in G_1$, довільні випадкові числа $y, \{h_i\}, i \in attributes$ і обчислюються $g^h, g^{h_i}$.
2. Генерація ключа: для атрибутів $S = \{A_1, A_2, \dots, A_n\}$ обчислюється $SK_s = \left\{g^{\left(y+h_{A_i}\right)}\right\} A_i \in S$.
3. Шифрування: для повідомлення M, політики доступу T і випадкового числа s: $CT = (C = M * e(g,g)^{ys},\ C` = g^s, \{C_i = (g^{h_i})^s\} i \in T$.
4. Розшифрування якщо атрибути S користувача задовольняють політику T, обчислюється:

$$M = \frac{C}{e(C`, SK_s)}, \tag{3.2}$$

### 3.1.5 JPBC

Java Pairing-Based Cryptography – це бібліотека для реалізації криптографічних алгоритмів, що базуються на технології білінійного парування. Парування, або білінійне парування, дозволяє перевести два елементи з однієї групи в елемент іншої групи [23]. JPBC є популярним інструментом для створення шифрувальних схем, зокрема атрибутивного шифрування(що використовується в нашому застосунку), систем електронних підписів і шифрування з відкритим текстом.

Парування використовує еліптичні криві, зокрема їх властивості, що дозволяють забезпечувати високу криптографічну стійкість при відносно невеликих розмірах ключів. Операція парування формально визначається як:

$$e: G_1 \times G_2 \rightarrow G_T, \tag{3.3}$$

де $G_1$ та $G_2$ – це адитивні групи на еліптичній кривій;

$G_T$ – мультиплікативна група.

Функція e має властивості білінійності, обчислюваності та ненульовості, які роблять її ідеальною для криптографії.

JPBC надає абстракції для роботи з еліптичними кривими, генерацією ключів та білінійними операціями. Це робить бібліотеку зручною для розробників, які бажають використовувати сучасні методи криптографії без детального занурення в складні математичні обчислення.

У нашому застосунку JPBC використовується для реалізації атрибутивного шифрування за методом CP-ABE. Цей підхід дозволяє шифрувати дані так, що доступ до них можна отримати лише за умови виконання певних умов (політики доступу). Наприклад, у нашому застосунку це перевірка атрибуту isAdmin і відповідності геолокації.

JPBC забезпечує генерацію приватних і публічних ключів на основі парування. Відповідно до методу CP-ABE, кожен користувач отримує приватний

ключ, що відповідає його атрибутам, наприклад, isAdmin = true або isAdmin = false. Шифрування даних здійснюється на основі політики доступу, визначеної в термінах цих атрибутів.

Для шифрування використовується публічний ключ системи та політика доступу: Ciphertext = E(Message,Policy).

Політика може бути виражена у вигляді формули, наприклад: (isAdmin∧ValidLocation) .

Це означає, що тільки адміністратор, який знаходиться у дозволеній геолокації, може отримати доступ до даних.

Коли користувач намагається отримати доступ до зашифрованих даних, його приватний ключ використовується разом із функцією розшифрування. JPBC виконує перевірку атрибутів та політики доступу: Message = D(Ciphertext, PrivateKey) .

Якщо умови не виконуються, розшифрування неможливе.

У нашому застосунку JPBC забезпечує потужний і гнучкий механізм шифрування даних на основі атрибутів. Це дозволяє гарантувати, що доступ до чутливої інформації можливий лише для користувачів із відповідними правами доступу та з урахуванням їхньої геолокації. Використання JPBC у поєднанні з BiometricPrompt та Room Persistence Library створює багаторівневу систему безпеки, яка відповідає сучасним вимогам до мобільних додатків.

### 3.2 Алгоритми роботи застосунку

Застосунок менеджер паролів на основі біометричної автентифікації користувачів розроблений з метою зберігання чутливої інформації у безпечному середовищі, зокрема паролів для різних вебсайтів. Основні алгоритми роботи застосунку охоплюють кілька ключових аспектів: автентифікацію користувачів, шифрування даних, взаємодію з базою даних та перевірку геолокації. Далі наведемо опис алгоритмів, що реалізовані в застосунку.

### 3.2.1 Загальний алгоритм роботи програми

З метою більш наглядної демонстрації принципу роботи створеного застосунку, створимо блок-схему, що відображає загальний хід роботи програми (Рис. 3.1):

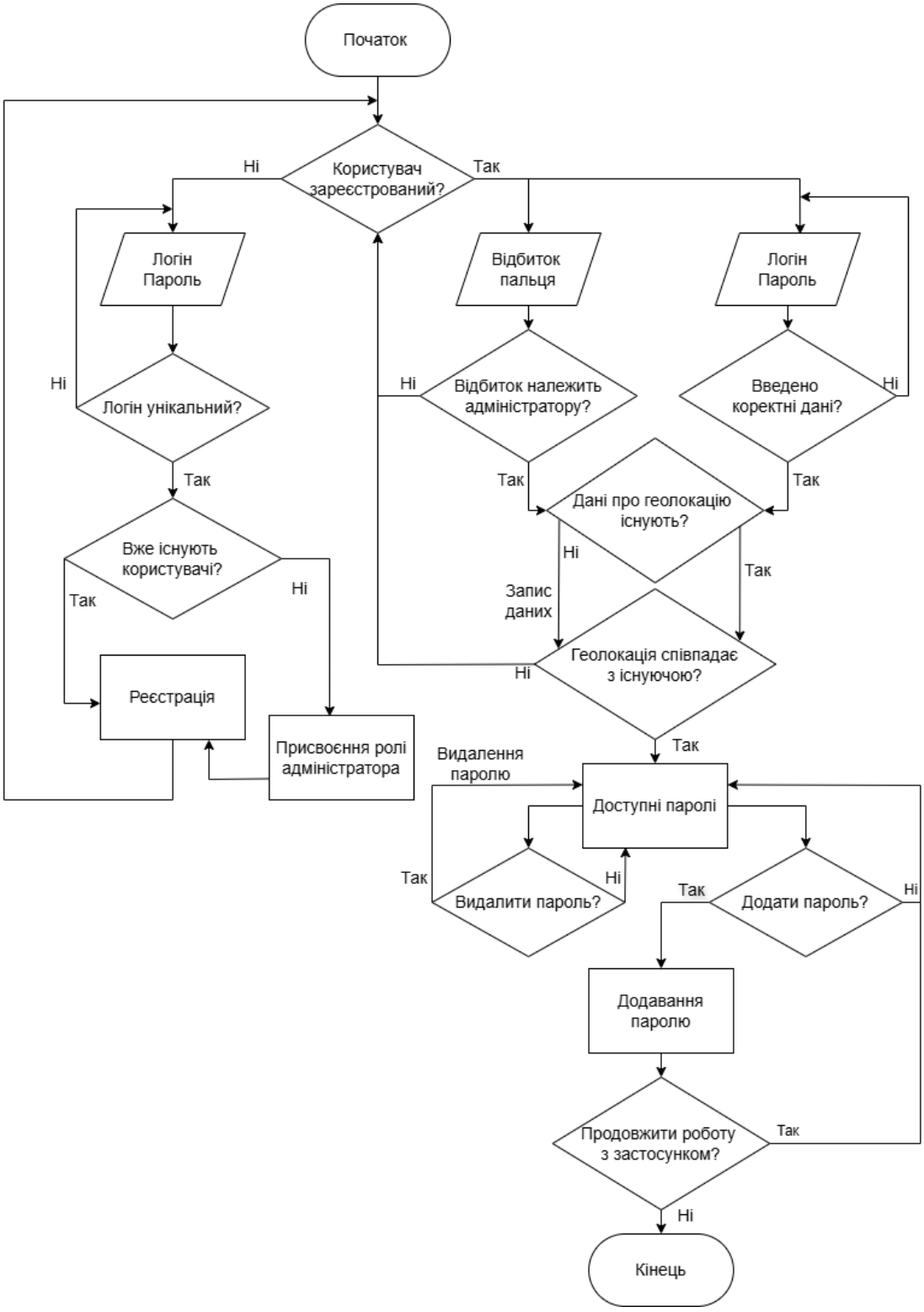


Рис. 3.1 Блок-схема роботи застосунку

### 3.2.2 Алгоритм автентифікації користувачів

Автентифікація є критичним етапом у забезпеченні безпеки застосунку. Вона включає використання біометричних даних та/або введення логіна та пароля.

- Перший крок. Під час першому запуску застосунку користувач повинен пройти процес реєстрації, де віе вводить логін та пароль. Після цього його дані зберігаються в базі даних.
- Другий крок. Коли користувач намагається увійти, застосунок перевіряє введені логін і пароль у базі даних за допомогою запиту до DAO. В якості альтернативи, користувач може автентифікуватися за рахунок біометричної автентифікації (за відбитком пальця).
- Третій крок. Якщо автентифікація через біометрію не вдалася або якщо користувач не бажає використовувати біометрію, застосунок дозволяє йому увійти через введення пароля.

### 3.2.3 Алгоритм перевірки геолокації

Геолокація є важливим компонентом безпеки, і застосунок використовує її для перевірки автентичності користувачів при кожному вході.

- Отримання геолокації. При першій авторизації користувач отримує свою геолокацію за допомогою FusedLocationProviderClient. Це забезпечує точні координати з використанням різних джерел, таких як GPS, Wi-Fi та мережеві бази станцій. Геолокація зберігається у базі даних для подальшого використання.
- Перевірка геолокації при наступних входах. Кожного разу, коли користувач намагається авторизуватися, застосунок порівнює поточну геолокацію з тією, що була зафіксована раніше. Якщо різниця в координатах значна, це може свідчити про можливу загрозу, тому доступ блокується.

### 3.2.4 Алгоритм роботи з базою даних

Room є основною бібліотекою для зберігання даних у застосунку. Вона працює через об'єкти DAO (Data Access Object) і використовує об'єктно-орієнтований підхід до зберігання даних.

- Додавання даних. Кожен новий запис про пароль, введений користувачем у форму, зберігається в базі даних через DAO. Це здійснюється за допомогою асинхронного виклику, щоб не блокувати головний потік користувача.
- Отримання даних. Для того, щоб отримати список паролів користувача, застосунок використовує LiveData з Room. Це дозволяє автоматично оновлювати UI, коли дані змінюються в базі даних.
- Видалення записів. Записи можуть бути видалені через відповідні функції DAO, що також працюють асинхронно, забезпечуючи швидку та ефективну роботу застосунку.

### 3.2.5 Алгоритм біометричної автентифікації

Для покращення безпеки застосунок використовує біометричну автентифікацію через BiometricPrompt API. Це дозволяє користувачам пройти ідентифікацію за допомогою відбитка пальця.

- Ініціалізація. Застосунок ініціалізує BiometricPrompt, який відкриває діалог для введення біометричних даних.
- Перевірка. Якщо біометричні дані коректні, користувач отримує доступ до застосунку, ідентифікація проходить без введення пароля. Якщо ж автентифікація не вдалася, застосунок повертається до введення пароля вручну.

### 3.2.6 Алгоритм атрибутивного шифрування за методом CP-ABE

Всі паролі в застосунку шифруються за допомогою CP-ABE, що дозволяє забезпечити доступ до даних тільки тим користувачам, які відповідають певним умовам.

- Шифрування. Паролі шифруються за допомогою публічного ключа, що дозволяє лише тим користувачам, які мають відповідні атрибути (наприклад, isAdmin = true), розшифровувати ці дані.
- Розшифрування. Кожен користувач має свій приватний ключ, який використовується для розшифрування даних. Якщо атрибути користувача не відповідають умовам, дані не можуть бути розшифровані.

### 3.2.7 Загальний алгоритм роботи програми

З метою більш наглядної демонстрації принципу роботи створеного застосунку, створимо блок-схему, що відображає загальний хід роботи програми (Рис. 3.1):

## 3.3 Висновки до третього розділу

В процесі виконання даного розділу було розглянуто технології, що використовувалися для створення застосунку менеджеру паролів на основі біометричної автентифікації користувачів. Окрім цього, було продемонстровано алгоритми роботи нашого застосунку.

Застосунок менеджеру паролів на основі біометричної автентифікації користувачів використовує низку алгоритмів з метою забезпечення високого рівня безпеки. Від автентифікації через біометрію та введення паролів до шифрування даних за методом CP-ABE та перевірки геолокації, всі ці процеси гарантують, що доступ до паролів матимуть тільки авторизовані користувачі з відповідними правами доступу. Кожен з цих алгоритмів працює ефективно та

безпечно, що дозволяє забезпечити конфіденційність і захист чутливої інформації.

В наступному розділі кваліфікаційної роботи роздивимось демонстрацію роботи застосунку на практиці, а також проведемо порівняльний аналіз з існуючими аналогами з метою встановлення доцільності використання створеного застосунку.

# РОЗДІЛ 4
# ПРОГРАМНИЙ ЗАСТОСУНОК МЕНЕДЖЕРА ПАРОЛІВ НА ОСНОВІ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

## 4.1 Опис середовища розробки

Застосунок менеджер паролів створено з допомогою мови програмування Kotlin.

Kotlin – це статично типізована мова програмування, яка працює на Java Virtual Machine (JVM), а також може компілюватися в JavaScript і створювати нативні додатки через Kotlin/Native. Мова була розроблена компанією JetBrains і офіційно підтримується Google для розробки Android-додатків.

Основні характеристики Kotlin включають:

Сумісність з Java: Kotlin є повністю сумісним з Java, що дозволяє використовувати бібліотеки та фреймворки Java без необхідності переписувати код. Kotlin має більш чистий синтаксис і підтримує сучасні концепції програмування.

Функціональність: Kotlin підтримує об'єктно-орієнтоване програмування (ООП) і функціональні можливості, такі як лямбда-вирази, високорівневі функції, замикання, та багато інших функцій, які дозволяють писати коротший і зрозуміліший код.

Безпечність від NPE: Kotlin надає вбудовану систему роботи з null-значеннями через систему nullable типів, що мінімізує ризик виникнення NullPointerException (NPE). Наприклад, щоб працювати з об'єктом, який може бути null, потрібно явно вказати тип як String?, що означає, що об'єкт може бути як рядком, так і null.

Іммутабельність: Kotlin заохочує використання іммутабельних (незмінних) об'єктів, що дозволяє знизити кількість помилок через зміну стану об'єктів.

Корутинні функції: Однією з найважливіших особливостей Kotlin є підтримка корутин – механізму для асинхронного програмування. Це дозволяє виконувати операції вводу/виводу або важкі обчислення в фоновому режимі без блокування основного потоку.

Лаконічність: Kotlin дозволяє писати код менш роздуто і в більш чистому вигляді, мінімізуючи шаблонний код, який часто трапляється в Java. Наприклад, у Kotlin можна обійтися без явного написання геттерів і сеттерів для полів класів завдяки використанню властивостей.

Множина функцій розширення: Kotlin дозволяє розширювати існуючі класи новими функціями без зміни їх коду, що дає можливість використовувати його для роботи з бібліотеками або фреймворками, не модифікуючи їх безпосередньо.

Завдяки своїй зручності, можливостям для підтримки сучасних підходів до програмування, а також тісній інтеграції з Java, Kotlin став однією з найпопулярніших мов для розробки Android-додатків.

Для запуску коду я використовував Android Studio.

Android Studio – це офіційне середовище розробки (IDE) для створення Android-додатків. Воно було розроблене компанією Google на основі популярного середовища JetBrains IntelliJ IDEA. Android Studio надає всі необхідні інструменти для розробки, тестування та відлагодження Android-додатків.

Основні особливості Android Studio:

1. Інтерфейс користувача: Android Studio має зручний та інтуїтивно зрозумілий інтерфейс для розробників. В ньому є вікна для роботи з кодом, перегляду ресурсів, дебаггінгу, а також для налаштування проектів і залежностей.
2. Підтримка Kotlin та Java: Android Studio підтримує як Kotlin, так і Java, що дозволяє вибирати мову програмування в залежності від переваг розробника чи вимог проекту. Kotlin стає стандартом для розробки Android-додатків, тому IDE має вбудовану підтримку для цієї мови,

надаючи корисні функції автозавершення, рефакторингу та перевірки помилок.

3. Інструменти для дизайну інтерфейсу: Android Studio включає потужні інструменти для створення графічного інтерфейсу користувача (UI). Це – розробка за допомогою XML у поєднанні з візуальним редактором, який дає можливість швидко створювати макети, перевіряти їх на різних екранах та роздільних здатностях, а також тестувати адаптивність додатків.
4. Підтримка емуляторів Android: Android Studio постачається з емуляторами для тестування додатків на різних пристроях і версіях Android. Це дозволяє розробникам перевіряти додатки без необхідності використання фізичного пристрою, що значно полегшує процес тестування і відлагодження.
5. Інструменти для профілювання та відлагодження: Android Studio має потужні інструменти для профілювання продуктивності додатків, моніторингу використання пам'яті та інших ресурсів. Вбудовані інструменти для відлагодження дозволяють швидко виявляти помилки у коді, контролювати виконання додатка і перевіряти різні аспекти його роботи.
6. Gradle: Android Studio використовує систему збірки Gradle для керування залежностями, збіркою та тестуванням проектів. Gradle дозволяє налаштувати складні процеси збірки, визначити різні типи збірок (наприклад, для дебагу чи релізу), а також інтегрувати з іншими інструментами та сервісами.
7. Підтримка Firebase: Android Studio має інтеграцію з Firebase – набором інструментів для створення, керування та моніторингу мобільних додатків. Це дає можливість швидко додавати такі функції, як аутентифікація, база даних у реальному часі, аналітика і багато іншого.
8. Засоби для тестування: Android Studio надає підтримку для автоматизованого тестування з використанням JUnit, Espresso та інших

фреймворків для перевірки функціональності додатків, що дозволяє забезпечити високу якість коду.

9. Інтеграція з Git: Android Studio вбудовує підтримку для системи контролю версій Git, що дозволяє зручно працювати з репозиторіями, здійснювати коміти, відслідковувати зміни та співпрацювати з іншими розробниками.

### 4.2 Демонстрація роботи програми

Розглянемо процес взаємодії користувача з програмою. При відкритті програми вас вітає стартове вікно менеджера паролів (рис 4.1):

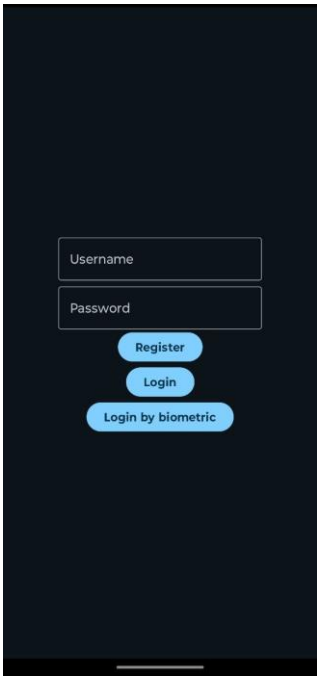


Рис. 4.1 Стартове вікно менеджера паролів

На цьому етапі ви маєте три варіанти вибору:

- Зареєструватися. Якщо у вас ще немає створеного акаунту – придумайте і введіть логін та пароль і натисніть кнопку «Register». Першому створеному акаунту буде присвоєно роль адміністратора. На цьому ж етапі відбувається збереження геолокації для подальших перевірок.

- Увійти. Якщо ви вже маєте створений акаунт – введіть логін та пароль і натисніть кнопку «Login» – ви перейдете до головного вікна менеджеру паролів.
- Увійти за відбитком пальця. Після натискання на цю кнопку відкриється форма для сканування відбитку пальця. Вхід за відбитком доступний лише для адміністратора.

Після успішної авторизації користувач переходить до головного вікна застосунку (рис. 4.2):

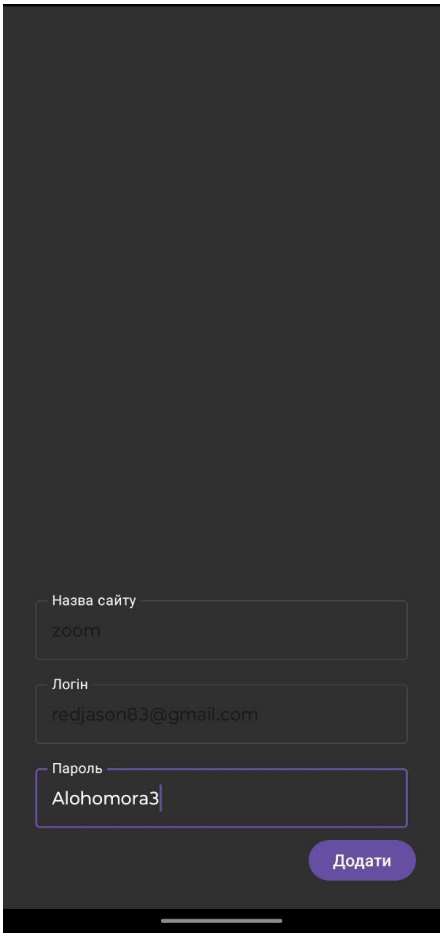


Рис. 4.2 Головний екран застосунку

На цьому етапі ви можете продивитися доступні вам створені записи, або додати нові, заповнивши відповідні поля та натиснувши кнопку «Додати». Створимо декілька записів для адміністратора (рис.4.3):

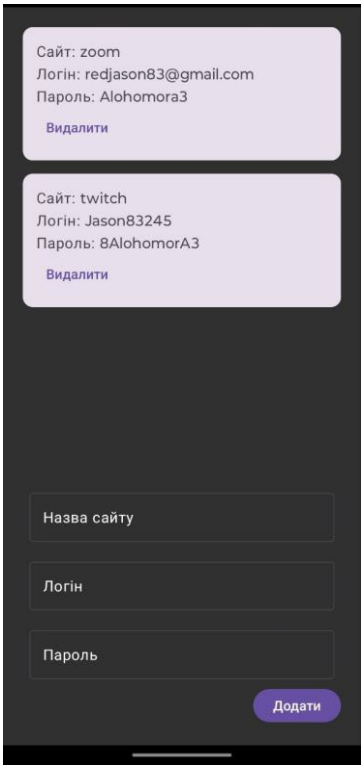


Рис. 4.3 Головний екран зі створеними записами

Для перевірки спробуємо авторизуватися на відповідних ресурсах за допомогою даних, збережених в менеджері паролів (рис. 4.4, 4.5):

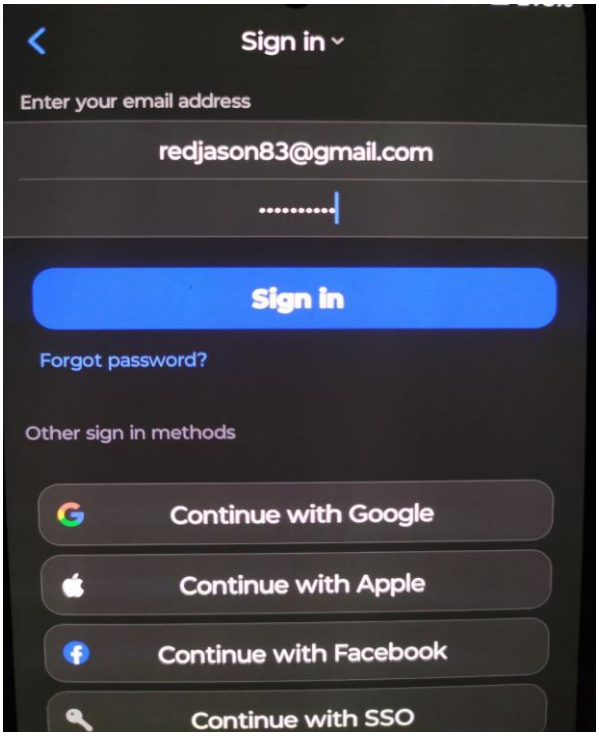


Рис. 4.4 Авторизація в Zoom

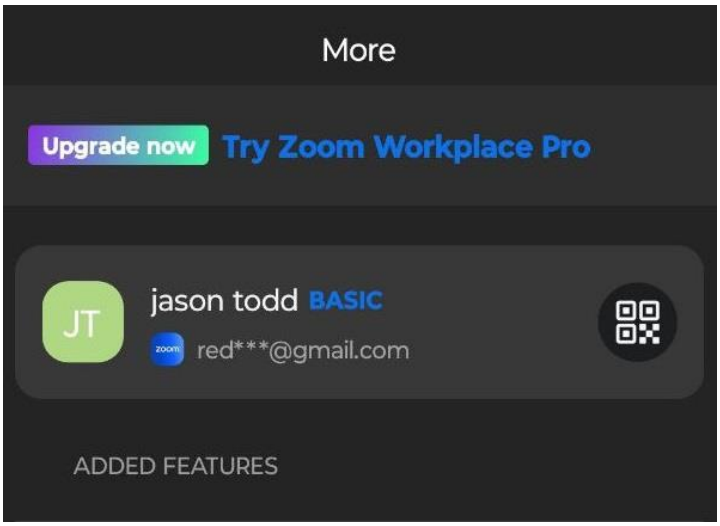


Рис. 4.5 Авторизація успішна

Далі, створимо нового користувача. В нього вже не буде прав адміністратора (рис 4.6):

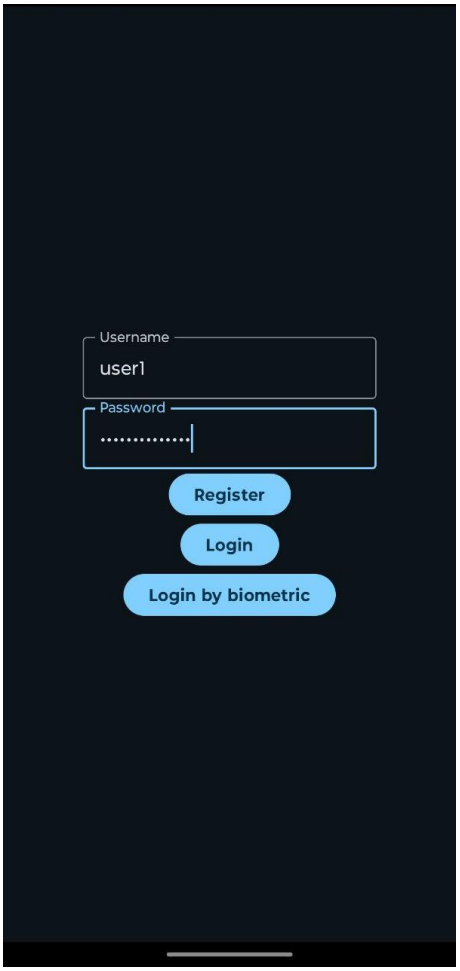


Рис. 4.6 Заповнення параметрів звичайного користувача

Оскільки цей користувач має доступ лише до своїх власних записів, основна сторінка застосунку в нього буде порожньою (рис. 4.7):

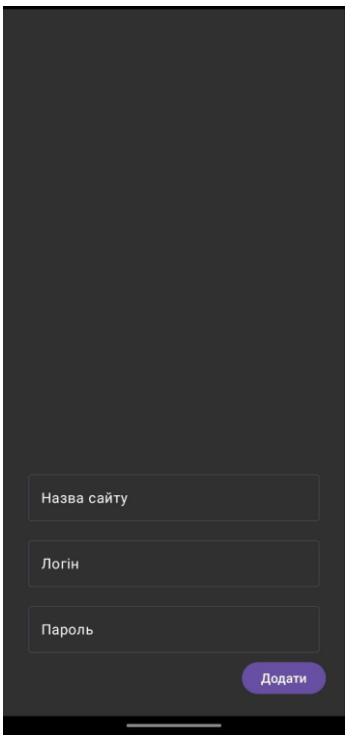


Рис. 4.7 Сторінка записів нового користувача

Для подальшої перевірки додамо цьому користувачу декілька записів з тими ж ресурсами, але іншими даними (рис. 4.8):

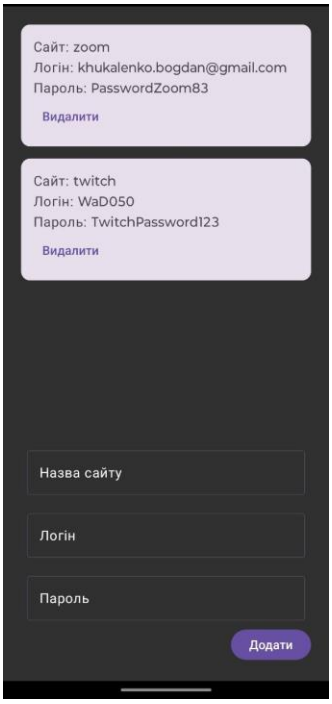


Рис. 4.8 Паролі звичайного користувача

Перевіримо авторизацію за вказаними користувачем даними (рис. 4.9):

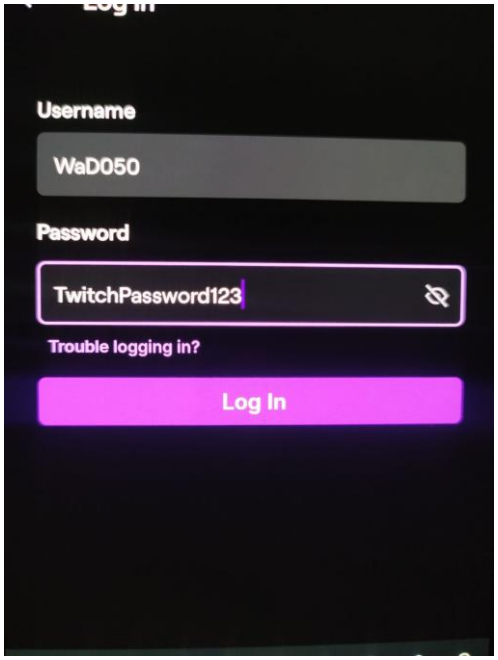


Рис. 4.9 Авторизація до ресурсу Twitch

Авторизація пройшла успішно (рис. 4.10):

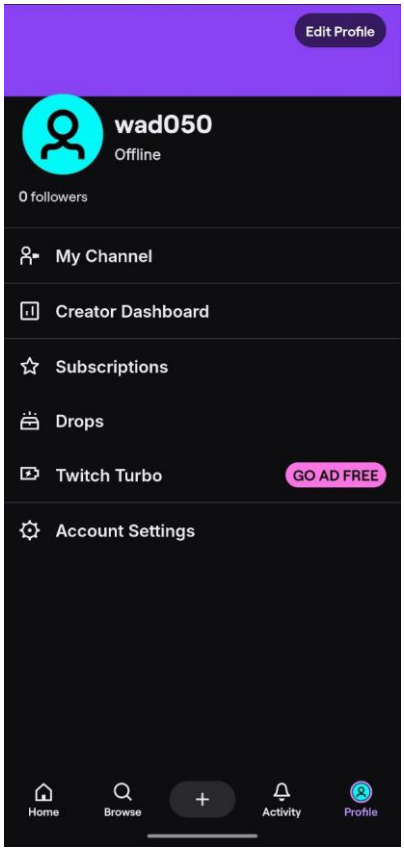


Рис. 4.10 Успішна авторизація на Twitch

Далі вийдемо з акаунту користувача та зайдемо знову до акаунту адміністратора. Для цього використаємо автентифікацію за відбитком пальця (рис. 4.11):

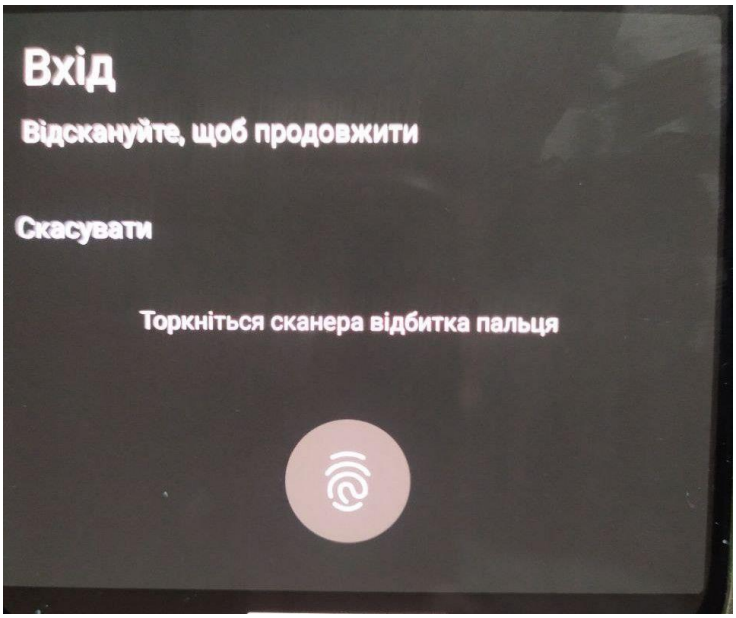


Рис. 4.11 Вікно входу за відбитком пальцю

В разі спроби авторизації за відбитком будь-кого окрім адміністратора автентифікація не буде успішною (рис. 4.12):

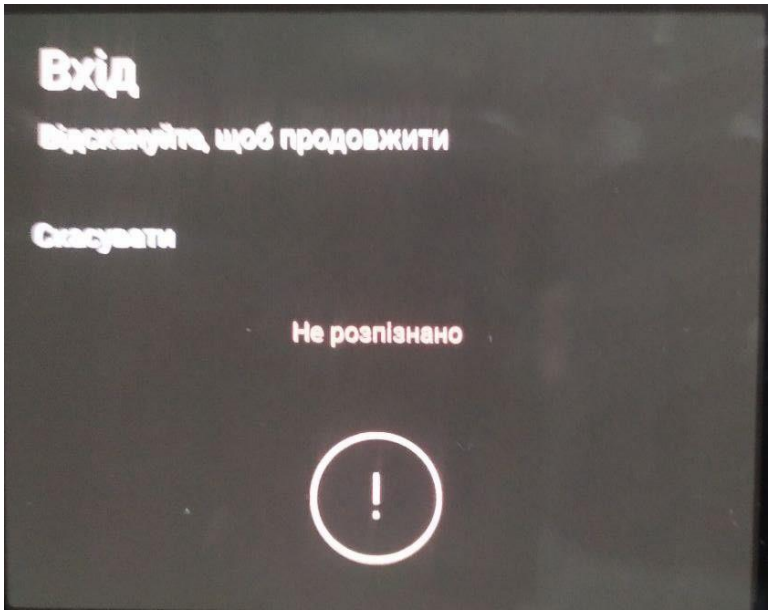


Рис. 4.12 Невдала спроба автентифікації

Після успішної автентифікації за відбитком адміністратор потрапляє на сторінку з доступними йому записами (рис. 4.13):

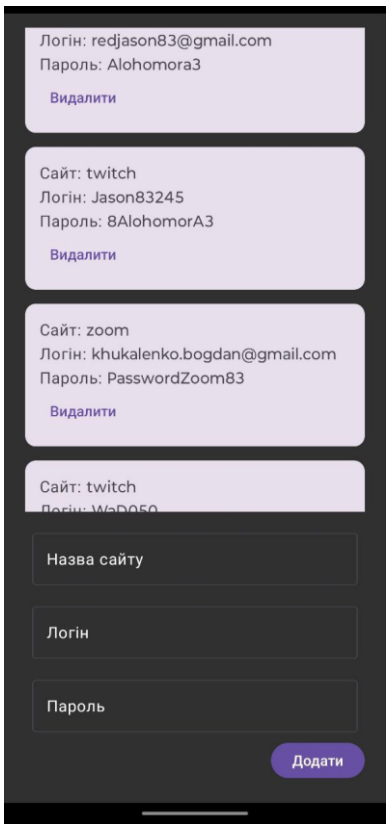


Рис. 4.13 Записи, доступні адміністратору

Як ми можемо побачити, адміністратор має змогу переглядати всі записи, створені користувачами.

Якщо спроба автентифікації відбуватиметься на відмінній від оригінальної геолокації – застосунок заблокує авторизацію та виведе повідомлення про помилку:

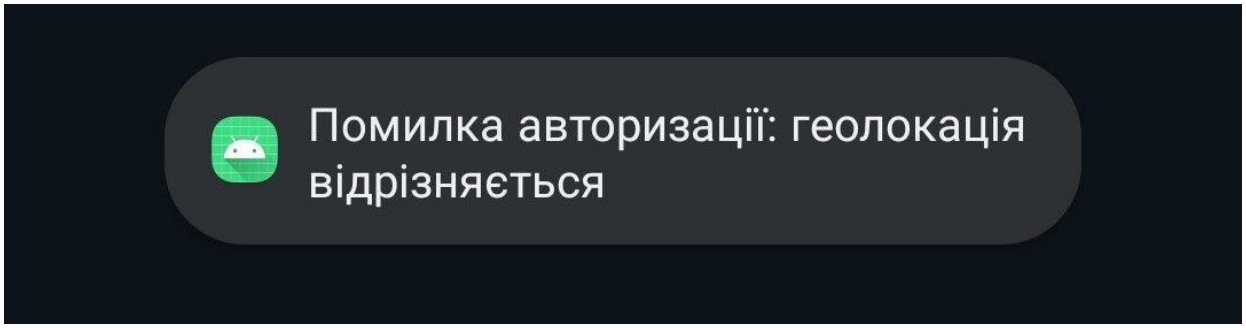


Рис. 4.14 Помилка авторизації при розбіжності геолокації

### 4.3 Оцінка ефективності та порівняння з існуючими системами

З метою винесення оцінки ефективності створеного додатку розглянемо наступні параметри:

1. Функціональність. Застосунок реалізує основні функції менеджера паролів:
   - Додавання та видалення записів із паролями та логінами.
   - Біометрична автентифікація та перевірка геолокації користувача при авторизації.
   - Шифрування паролів за допомогою методу CP-ABE для забезпечення конфіденційності.
   - Управління користувачами з різними рівнями доступу (адміністратор, звичайний користувач).

   Функціональність відповідає вимогам менеджера паролів, і додаткові функції, такі як перевірка геолокації та атрибутивне шифрування, значно підвищують безпеку та зручність використання.
2. Користувацький інтерфейс. Графічний інтерфейс застосунку забезпечує простоту використання і наочність, що покращує взаємодію користувача з програмою.
3. Продуктивність. У застосунку застосовуються корутини для асинхронного доступу до бази даних, що допомагає уникнути блокування головного потоку під час роботи з даними. Використання LiveData забезпечує реактивність інтерфейсу при зміні даних, а перевірка геолокації виконується без блокування UI потоку. Проте варто зазначити, шо додавання складних операцій, таких як атрибутивне шифрування (CP-ABE) та перевірка геолокації, може створювати додаткове навантаження при великій кількості користувачів та великих обсягах даних.
4. Безпека. Застосунок має кілька рівнів безпеки:
   - Використання біометричної автентифікації та паролів.

- Шифрування паролів за допомогою атрибутивного шифрування (CP-ABE), що дозволяє контролювати доступ до даних за атрибутами.
- Перевірка геолокації для уникнення несанкціонованого доступу в разі значних змін у місцезнаходженні.

5. Оновлюваність і підтримка. Оновлюваність цього застосунку забезпечена через:
   - Використання архітектури MVVM та підтримку ViewModel, що дозволяє спростити оновлення бізнес-логіки без значних змін у UI.
   - Використання бібліотек, які активно підтримуються (наприклад, Room, Coroutines).
6. Сумісність і залежності. Застосунок залежить від кількох бібліотек, зокрема:
   - Room: для роботи з базою даних.
   - Coroutines: для асинхронної роботи.
   - JPBC: для шифрування CP-ABE.
   - AndroidX: для підтримки сучасних компонентів UI.

   Вцілому, залежності з сучасними бібліотеками забезпечують хорошу сумісність з Android, але наявність специфічних бібліотек, таких як JPBC, може спричиняти проблеми з сумісністю при оновленнях або у разі змін у підтримці бібліотек.

Загалом, застосунок має сильну функціональність і безпеку, але є певні виклики в плані масштабованості, оновлюваності та сумісності через використання складних алгоритмів шифрування та специфічних бібліотек. Системи шифрування та геолокації можуть вимагати додаткових налаштувань та оптимізації для забезпечення ефективної роботи при зростанні навантаження.

Проведемо порівняння запропонованого в ході цієї роботи рішення з менеджерами паролів, які були розглянуті у першому пункті третього розділу. Для цього зробимо нову порівняльну таблицю з використанням тих самих критеріїв (табл. 4.1):

Таблиця 4.1

Порівняння існуючих рішень з менеджменту паролів з розробленим застосунком

| Менеджер паролів | LastPass | 1Password | Bitwarden | Dashlane | Keeper | Розроблений застосунок |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Функціональність | Зберігання паролів, автоматичне заповнення, генерація паролів, синхронізація (у платному плані). | Сильна підтримка сімейних і бізнес-акаунтів, генерація паролів, збереження документів, поділ на "сховища". | Синхронізація на всіх пристроях, генерація паролів, збереження нотаток. | Зберігання паролів, VPN у преміум-плані, автоматична зміна паролів, перевірка слабких паролів | Фокус на безпеку, підтримка збереження файлів, історія записів, приватні повідомлення. | Зберігання паролів, біометрична авторизація, перевірка геолокації. |
| Ціна | Безкоштовний план, Premium $3/місяць. | Безкоштовний план, Premium $10/рік. | Безкоштовний план, Premium $10/рік. | Безкоштовний план, Premium $10/рік. | Від $2.91/місяць, окремо захист файлів. | Безкоштовний. |
| Інтерфейс | Інтуїтивний, але були скарги на перевантаженість інтерфейсу. | Чистий, зручний, підтримує різні платформи. | Простіший інтерфейс, менш сучасний вигляд.. | Сучасний, але іноді складний у налаштуванні. | Сучасний, фокус на безпеку, доступність інтеграції | Функціональний інтерфейс для додавання записів, та |

Закінчення таблиці 4.1

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| | | | | | з іншим ПЗ. | автентифікації за відбитком пальця. |
| Рівень захисту при втраті пристрою | 2FA, можливість відновлення облікового запису через головний пароль. | 2FA, можливість встановлення параметра "Автоблокування" | Підтримує 2FA | 2FA, повідомлення про компрометацію даних. | Підтримка 2FA, віддалене очищення даних, аудит безпеки | Біометрична автентифікація, шифрування паролів за допомогою CP-ABE, перевірка геолокації при авторизації. |
| Відкритість коду | Закритий код | Закритий код | Відкритий код | Закритий код | Закритий код | Відкритий код |
| Історія вразливостей | Є історія витоків (2022). | Були випадки порушення безпеки (2023). | Без історії серйозних витоків | Відомі незначні вразливості. | Жодних відомих витоків. | Немає відомих вразливостей (нова система). |
| Алгоритм шифрування | AES-256 | AES-256 | AES-256 | AES-256 | AES-256 | CP-ABE |
| Вірогідність злому | $1.84 \times 10^{-14}$ | $1.84 \times 10^{-14}$ | $1.84 \times 10^{-14}$ | $1.84 \times 10^{-14}$ | $1.84 \times 10^{-14}$ | $5.88 \times 10^{-48}$ |

Окрім попередніх критерії оцінки було додано ще один – вірогідність злому. Для того щоб провести оцінку проведемо розрахунки вірогідності:

1. Менеджер паролів з AES. Найбільш вразливим параметром в системах, що захищені з допомогою алгоритму AES є майстер-пароль, компрометація якого надає повний доступ до захищеної інформації. Отже, вірогідність злому розрахуємо за формулою:

$$P_{MP} = \frac{1}{N_{CS}^{L_{MP}}} = \frac{1}{62^8} = 1.84 \times 10^{-14}, \quad (4.1)$$

де $L_{MP}$ – довжина майстер-паролю,

$N_{CS}$ – простір можливих символів

2. Для того, щоб обійти створену нами систему зловмисникам необхідно обійти CP-ABE шифрування і підробити атрибути, підробити біометричні дані та змоделювати геолокацію. Вірогідність злому розрахуємо за формулою:

$$P_{Total} = P_{ABE} \times P_{Bio} \times P_{Geo} = 2.94 \times 10^{-39} \times 2 \times 10^{-5} \times 1 \times 10^{-4} = 5.88 \times 10^{-48}, \quad (4.2)$$

Оскільки розроблений застосунок є новим та невеликим рішенням, йому важко конкурувати з лідерами індустрії, котрі розроблялися протягом довгого часу великими командами спеціалістів, проте він все ж має перелік якостей, що позитивно виділяють його на фоні аналогів, попри невеликий список функцій та простий інтерфейс. До цих переваг можемо віднести:

- Відкритість коду**.** Застосунок має відкритий код, що дозволяє користувачам перевіряти, оновлювати та покращувати програму. Це підвищує прозорість і дозволяє залучати спільноту розробників до внесення змін, що значно підвищує рівень безпеки і довіри до застосунку. На відміну від закритих кодів у таких застосунках, як

LastPass або 1Password, відкритий код дає користувачам більше контролю.

- Біометрична автентифікація. Застосунок підтримує біометричну автентифікацію за відбитком пальця, що забезпечує більш зручний і безпечний спосіб авторизації порівняно з більшістю конкурентів, які в основному використовують лише пароль.
- Атрибутивне шифрування CP-ABE. Використання атрибутивного шифрування за методом CP-ABE дає додатковий рівень безпеки, дозволяючи регулювати доступ до паролів на основі атрибутів користувача. Це підвищує конфіденційність і контроль над доступом у порівнянні з іншими менеджерами паролів.
- Перевірка геолокації. Порівняно з конкурентами, застосунок може здійснювати перевірку геолокації користувача під час авторизації, що дозволяє виявляти підозрілі спроби доступу з нових або незвичних місць. Це значно підвищує рівень безпеки, що важливо в умовах сучасних загроз безпеці.

### 4.4 Висновки до четвертого розділу

У цьому розділі розглянуто розробку інноваційного програмного забезпечення для менеджера паролів, який використовує передові методи захисту, зокрема атрибутивне шифрування (CP-ABE) та біометричну автентифікацію. Цей застосунок має значний потенціал завдяки своїй здатності інтегрувати сучасні технології безпеки та пропонувати користувачам зручний інтерфейс для керування паролями.

Основною перевагою розробленого рішення є використання CP-ABE для забезпечення більш гнучкого контролю доступу, де доступ до збережених даних регулюється через атрибути користувача, такі як статус адміністратора та геолокація. Це значно підвищує рівень захисту та контролю над даними,

оскільки лише авторизовані користувачі можуть отримувати доступ до конфіденційної інформації.

Хоча для більшості користувачів це може не бути оптимальним варіантом в порівнянні з комерційними рішеннями, такими як 1Password або KeePass, будь який програмний застосунок створюється під свого користувача та має ряд своїх переваг та недоліків, отже важливо врахувати сильні сторони розробленого застосунку та вирішувати, чи він задовольняє потреби конкретних категорій споживачів.

Незважаючи свої недоліки, застосунок буде ідеальним вибором для тих користувачів, які ставлять на перший план безпеку та прозорість. Вони готові прийняти певні компроміси в швидкості та зручності заради підвищеного рівня захисту своїх даних. Оскільки відкритий код дозволяє розвивати програму та адаптувати її під різні сценарії, це рішення є універсальним для користувачів, які цінують безпеку та приватність.

# ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було розглянуто цінність інформації в сучасну технологічну епоху та визначено, що в умовах постійного зростання рівня загроз та викликів в сфері кібербезпеки, питання безпеки інформації стає дедалі актуальнішим. Було висвітлено той факт, що цінність інформації залежить від характеристик, якими вона володіє, та розглянуто три основні характеристики: конфіденційність, цілісність та доступність.

Окрім цього, було наголошено на важливості надійності паролів в контексті захисту особистих даних користувачів та розглянуто дослідження, що демонструє найпопулярніші паролі серед користувачів. Основуючись на результатах дослідження ми дійшли висновку, що переважна більшість паролів користувачів є надзвичайно простими та ненадійними, неспроможними надати адекватний рівень безпеки інформації. Причиною цьому служить те, що надійні паролі являються надто складними для запам'ятовування в багатьох програмах. Маючи вибір, користувачі зазвичай створюють дуже спотворений розподіл вибору паролів, що відображає загальну семантику, яка полегшує запам'ятовування (наприклад, поєднання слова або послідовності чисел із особливим значенням).

Було встановлено, що оптимальним вирішенням встановленої проблеми є використання менеджера паролів. Вони дозволяють користувачам зберігати паролі в безпеці, уникати фішингових атак і спростити процес управління обліковими записами в різних сервісах.

Також було проведено аналіз сучасних методів автентифікації, зокрема, їх три основні характеристики: ступінь автоматизації, пріоритет використання та використовуваний фактор автентифікації.

Засновуючись на результатах проведеного аналізу, біометричну автентифікацію було обрано як оптимальний метод для використання в застосунку менеджера паролів.

Також нами було проаналізовано методи біометричної автентифікації. З розглянутих варіантів метод ідентифікації за відбитком пальця було обрано як оптимальний для застосування в застосунку менеджері паролів.

Окрім цього, було описано найбільш вживані алгоритми порівняння відбитків пальців. Метод порівняння за мінюціями виявився найефективнішим з огляду на низку ключових переваг над іншими методами. Порівняння за мінюціями базується на обмеженій кількості унікальних, що дозволяє зменшити обсяг обчислень і прискорити процес. Алгоритм мінюцій адаптується до трансляції, обертання та масштабування зображення завдяки нормалізації координат мінюцій. Також, метод мінюцій може працювати навіть із частково пошкодженими відбитками, оскільки йому потрібні лише локалізовані точки розгалужень і закінчень.

До того ж, було детально розглянуто технології, що використовувалися для створення застосунку менеджеру паролів на основі біометричної автентифікації користувачів. Окрім цього, було продемонстровано алгоритми роботи нашого застосунку. Застосунок менеджеру паролів на основі біометричної автентифікації користувачів використовує низку алгоритмів з метою забезпечення високого рівня безпеки. Від автентифікації через біометрію та введення паролів до шифрування даних за методом CP-ABE та перевірки геолокації, всі ці процеси гарантують, що доступ до паролів матимуть тільки авторизовані користувачі з відповідними правами доступу. Кожен з цих алгоритмів працює ефективно та безпечно, що дозволяє забезпечити конфіденційність і захист чутливої інформації.

Для визначення доцільності використання створеного застосунку його було протестовано та проведено порівняння з існуючими аналогами в цій сфері. В результаті проведеного порівняння ми дійшли висновку, що сильні сторони застосунку – функціональність і безпека, але він має певні виклики в плані масштабованості, оновлюваності та сумісності через використання складних алгоритмів шифрування та специфічних бібліотек. Зокрема, системи

шифрування та геолокації можуть вимагати додаткових налаштувань та оптимізації для забезпечення ефективної роботи при зростанні навантаження.

При порівнянні з існуючими аналогами, основною перевагою розробленого рішення виявилось використання CP-ABE для забезпечення більш гнучкого контролю доступу, де доступ до збережених даних регулюється через атрибути користувача, такі як статус адміністратора та геолокація. Це значно підвищує рівень захисту та контролю над даними, оскільки лише авторизовані користувачі можуть отримувати доступ до конфіденційної інформації.

Отже, незважаючи свої недоліки, створений застосунок буде ідеальним вибором для тих користувачів, які ставлять на перший план безпеку та прозорість. Вони готові прийняти певні компроміси в швидкості та зручності заради підвищеного рівня захисту своїх даних. Оскільки відкритий код дозволяє розвивати програму та адаптувати її під різні сценарії, це рішення є універсальним для користувачів, які цінують безпеку та приватність.

В результаті виконання кваліфікаційної роботи отримані наступні результати:

1)Проаналізовано сучасні методи автентифікації та біометричної ідентифікації та на основі результату проведеного аналізу обрано оптимальні технології для вирішення задачі з розробки менеджера паролів.

2)Розроблено програмний застосунок менеджер паролів на основі біометричної автентифікації з використанням атрибутивного шифрування та геолокації, що дало змогу забезпечити підвищений рівень захисту даних користувачів.

3)Проведено тестування розробленого застосунку, на основі якого визначено що використання застосунку є доцільним для розв’язання задачі з управління паролями і підвищення безпеки користувачів завдяки використанню атрибутивного шифрування та геолокації.

## СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Whitman M. E., Mattord H. J. Principles of Information Security. Cengage Learning, 2021. 11 p.

2. NordPass: “Top 200 Most Common Passwords”. URL: https://nordpass.com/most-common-passwords-list/

(дата звернення: 12.10.2024).

3. Das A., Bonneau J., Caesar M., Borisov N., Wang, X. The Tangled Web of Password Reuse. Network and Distributed System Security Symposium (NDSS), 2014. 2 p.

4. H. Padalia, H. Patel, A. Deshmukh, M. Patail, A. Kumar, N. Kumar Nrip. A Study on Password Manager: Users’ Perspective, 2023.

5. S. Sharma. A Comprehensive Study of Cryptographic Hash Functions, 2024

6. R. Ayyagari, J. Lim, O. Hoxga. Why Do Not We Use Password Managers? A Study on the Intention to Use Password Managers, 2019.

7. Клопотовський Д. О., Писаренко Л.Д. Класифікація механізмів аутентифікації користувачів і їх огляд, 2017.

8. J. Ashbourne. Biometrics: Advanced Identity Verification, 2000.

9. Zhou L., Vu M. T., Oechtering T. J., Skoglund M. Two-Stage Biometric Identification Systems without Privacy Leakage, 2021.

10. Maltoni D., Maio D., Jain A. K., Prabhakar S. Handbook of Fingerprint Recognition, 2009.

11. Zhang D. Biometric Solutions: For Authentication in an e-World. Springer Science & Business Media, 2013.

12. Jain A., Flynn P., Ross A. Handbook of Biometrics, 2007.

13. Висоцька О.О. Моніторинг роботи користувачів комп’ютерних систем за допомогою технологій розпізнавання за клавіатурним почерком / О.О. Висоцька// Моделювання та інформаційні технології. Збірник наукових праць

інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України. – К.:ІПМЕ, 2018. – Вип. 84. – С. 119-125.

14. Al-Rousan M., Benedetto I. A Comparative Analysis of Biometrics Types: Literature Review, 2020.

15. Gonzalez R., Woods R. Digital Image Processing, 1992.

16. Stallings W. Cryptography and Network Security: Principles and Practice (7th ed.). Pearson Education, 2017.

17. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms (3rd ed.). MIT Press, 2009.

18. Chaudhuri P. Cryptography and Security: From Theory to Practice. Wiley, 2019.

19. Shahzad M., Khan M. Advanced Cryptography and Secure Communication: Theory and Applications. Springer, 2017.

20. Gonzalez M. Mastering Android Database Development with Kotlin (1st ed.). Packt Publishing, 2019.

21. Doss D. Android Security Internals: An In-Depth Guide to Android's Architecture, Components, and Security Features. No Starch Press, 2020.

22. Bertoni G., Canetti R. Pairing-Based Cryptography: 6th International Conference, Pairing 2008 (Lecture Notes in Computer Science, Vol. 5209). Springer, 2008.

23. Hämmerle H. A. An Introduction to Pairing-Based Cryptography, 2005.

# ДОДАТКИ

## Код застосунку

MainActivity.kt

```
package com.example.passwordmanager

import …
class MainActivity : AppCompatActivity() {
    private lateinit var viewModel: MainViewModel
    private val promptManager by lazy {
        BiometricPromptManager(this)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        viewModel = ViewModelProvider(this,
MainViewModelFactory(application)).get(MainViewModel::class.java)
        setContent {
            val context = LocalContext.current
            PasswordManagerTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    val BiometricResult by
promptManager.promptResults.collectAsState(
                        initial = null
                    )
                    val enrollLauncher = rememberLauncherForActivityResult(
```

```
            contract = ActivityResultContracts.StartActivityForResult(),
            onResult = {
                println("Activity result: $it")
            }
        )
        LaunchedEffect(BiometricResult) {
            if (BiometricResult is
BiometricPromptManager.BiometricResult.AuthenticationNotSet) {
                if (Build.VERSION.SDK_INT >= 30) {
                    val enrollIntent =
Intent(Settings.ACTION_BIOMETRIC_ENROLL).apply {
                        putExtra(

Settings.EXTRA_BIOMETRIC_AUTHENTICATORS_ALLOWED,
                            BIOMETRIC_STRONG or
DEVICE_CREDENTIAL
                        )
                    }
                    enrollLauncher.launch(enrollIntent)
                }
            }
        }
        Column(
            modifier = Modifier.fillMaxSize(),
            verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally
        ) {

            var username by remember { mutableStateOf("") }
            var password by remember { mutableStateOf("") }
```

```
                val context = LocalContext.current

                Column(
                    modifier = Modifier.fillMaxSize(),
                    verticalArrangement = Arrangement.Center,
                    horizontalAlignment = Alignment.CenterHorizontally
                ) {
                    OutlinedTextField(
                        value = username,
                        onValueChange = { username = it },
                        label = { Text("Username") }
                    )
                    OutlinedTextField(
                        value = password,
                        onValueChange = { password = it },
                        label = { Text("Password") },
                        visualTransformation =
PasswordVisualTransformation()
                    )
                    Button(onClick = {
                        CoroutineScope(Dispatchers.Main).launch {
                            val existingAdmin = viewModel.allUsers.value
                            val newUser = User(
                                username = username,
                                password = password,
                                isAdmin =  existingAdmin.isNullOrEmpty()
                            )
                            viewModel.insertUser(newUser)
                            viewModel.getAllUsers()
                            val intent = Intent(context,
```

```
HomeActivity::class.java).apply {
                        putExtra("PASSWORD", password)
                        putExtra("USERNAME", username)
                    }
                    context.startActivity(intent)
                }
            }) {
                Text("Register")
            }
            Button(onClick = {
                CoroutineScope(Dispatchers.Main).launch {
                    val user = viewModel.authenticateUser(username,
password)
                    if (user != null) {
                        val intent = Intent(context,
HomeActivity::class.java).apply {
                            putExtra("PASSWORD", password)
                            putExtra("USERNAME", username)
                        }
                        context.startActivity(intent)
                    } else {
                        // Show error
                    }}
            }) {
                Text("Login")
            }
            Button(onClick = {
                promptManager.showBiometricPrompt(
                    title = "Вхід",
                    description = "Відскануйте, щоб продовжити"
```

```
                )
            }
            ) {
                Text(text = "Login by biometric")
            }
        }



        BiometricResult?.let { result ->
            Text(
                text = when (result) {
                    is
BiometricPromptManager.BiometricResult.AuthenticationError -> {
                        result.error
                    }

BiometricPromptManager.BiometricResult.AuthenticationFailed -> {
                        "Authentication failed"
                    }

BiometricPromptManager.BiometricResult.AuthenticationNotSet -> {
                        "Authentication not set"
                    }

BiometricPromptManager.BiometricResult.AuthenticationSuccess -> {
                        "Authentication success"
```

```
                    }


BiometricPromptManager.BiometricResult.FeatureUnavailable -> {
                        "Feature unavailable"
                    }


BiometricPromptManager.BiometricResult.HardwareUnavailable -> {
                        "Hardware unavailable"
                    }
                }
            )
            if (result ==
BiometricPromptManager.BiometricResult.AuthenticationSuccess) {

                context.startActivity(Intent(context,
HomeActivity::class.java))
            }

        }
    }
  }
}
```

BiometricPromptManager.kt

```
package com.example.passwordmanager

import …

class BiometricPromptManager(
    private val activity: AppCompatActivity
) {
    private val resultChannel = Channel<BiometricResult>()
    val promptResults = resultChannel.receiveAsFlow()

    fun showBiometricPrompt(
        title: String,
        description: String
    ) {
        val manager = BiometricManager.from(activity)
        val authenticators = if(Build.VERSION.SDK_INT >= 30) {
            BIOMETRIC_STRONG or DEVICE_CREDENTIAL
        } else BIOMETRIC_STRONG

        val promptInfo = PromptInfo.Builder()
            .setTitle(title)
            .setDescription(description)
            .setAllowedAuthenticators(authenticators)

        if(Build.VERSION.SDK_INT < 30){
            promptInfo.setNegativeButtonText("Cancel")
        }
```

```
        when(manager.canAuthenticate(authenticators)) {
            BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE ->
{
                resultChannel.trySend(BiometricResult.HardwareUnavailable)
                return
            }
            BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE ->{
                resultChannel.trySend(BiometricResult.FeatureUnavailable)
                return
            }
            BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED -> {
                resultChannel.trySend(BiometricResult.AuthenticationNotSet)
                return
            }
            else -> Unit
        }

        val prompt = BiometricPrompt(
            activity,
            object : BiometricPrompt.AuthenticationCallback() {
                override fun onAuthenticationError(errorCode: Int, errString:
CharSequence) {
                    super.onAuthenticationError(errorCode, errString)

resultChannel.trySend(BiometricResult.AuthenticationError(errString.toStri
ng()))
                }

                override fun onAuthenticationSucceeded(result:
BiometricPrompt.AuthenticationResult) {
```

```
                super.onAuthenticationSucceeded(result)
                resultChannel.trySend(BiometricResult.AuthenticationSuccess)
            }

            override fun onAuthenticationFailed() {
                super.onAuthenticationFailed()
                resultChannel.trySend(BiometricResult.AuthenticationFailed)
            }
        }
    )
    prompt.authenticate(promptInfo.build())
}

sealed interface BiometricResult{
    data object HardwareUnavailable: BiometricResult
    data object FeatureUnavailable: BiometricResult
    data class AuthenticationError(val error: String): BiometricResult
    data object AuthenticationFailed: BiometricResult
    data object AuthenticationSuccess: BiometricResult
    data object AuthenticationNotSet: BiometricResult
}
}
```

MainViewModel.kt

```
package com.example.passwordmanager

import android.app.Application
import androidx.lifecycle.AndroidViewModel
```

```
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch


class MainViewModel(application: Application):
AndroidViewModel(application) {
    private val userDao = AppDatabase.getDatabase(application).userDao()
    val _allUsers = MutableLiveData<List<User>> ()
    val allUsers : LiveData<List<User>> = _allUsers
    private val _currentUser = MutableStateFlow<User?>(null)
    val currentUser: StateFlow<User?> = _currentUser
    init {
        getAllUsers()
    }fun insertUser(user: User) = viewModelScope.launch {
        userDao.insertUser(user)
    }
    fun authenticateUser(username: String, password: String) =
viewModelScope.launch {
        _currentUser.value = userDao.authenticateUser(username, password)
    }
    fun getAllUsers()  {
        viewModelScope.launch {
            _allUsers.postValue(userDao.getAllUsers())
        }
```

```
    }
}
```

PasswordListScreen.kt

```
package com.example.passwordmanager
import…
@Composable
fun PasswordListScreen(viewModel: PasswordViewModel, username:
String, password: String){

    viewModel.getCurrentUser(username, password)
val user by viewModel.currentUser.observeAsState()
    user?.userId?.let { viewModel.getEntriesForUser(it) }
    val entries by viewModel.allEntries.observeAsState(emptyList())
    Column (modifier = Modifier.fillMaxSize().padding(16.dp)){
        LazyColumn(modifier = Modifier.weight(1f)) {
            entries?.size?.let {
                items(it){ entry ->
                    PasswordItem(entry = entries[entry], onDelete =
{viewModel.delete(it)})
                }
            }
        }
        AddEntryForm(onAddEntry = {newEntry ->
            viewModel.insert(newEntry)
        }, user?.userId)
    }
}
@Composable
```

```
fun PasswordItem(entry: PasswordEntry, onDelete: (PasswordEntry) ->
Unit) {
    Card(modifier = Modifier.fillMaxWidth().padding(8.dp)) {
        Column(modifier = Modifier.padding(16.dp)){
            Text(text = "Сайт: ${entry.siteName}")
            Text(text = "Логін: ${entry.login}")
            Text(text = "Пароль: ${entry.password}")
            TextButton(onClick =  { onDelete(entry) } ) {
                Text("Видалити")
            }
        }

    }
}
@Composable
fun AddEntryForm(onAddEntry: (PasswordEntry) -> Unit, userID: Int?) {
    var siteName by remember { mutableStateOf("") }
    var login by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }

    Column(modifier = Modifier.fillMaxWidth().padding(8.dp)){
        OutlinedTextField(
            value = siteName,
            onValueChange = {siteName = it},
            label = { Text("Назва сайту", color = Color.White) },
            modifier = Modifier.fillMaxWidth().padding(8.dp),
            colors = TextFieldDefaults.colors(
                focusedTextColor =  Color.White,
                focusedContainerColor = Color.Transparent,
                unfocusedContainerColor = Color.Transparent)
```

```
        )
        OutlinedTextField(
            value = login,
            onValueChange = {login = it},
            label = { Text("Логін", color = Color.White) },
            modifier = Modifier.fillMaxWidth().padding(8.dp),
            colors = TextFieldDefaults.colors(
                focusedTextColor =  Color.White,
                focusedContainerColor = Color.Transparent,
                unfocusedContainerColor = Color.Transparent)
        )
        OutlinedTextField(
            value = password,
            onValueChange = {password = it},
            label = { Text("Пароль", color = Color.White) },
            modifier = Modifier.fillMaxWidth().padding(8.dp),
            colors = TextFieldDefaults.colors(
                focusedTextColor =  Color.White,
                focusedContainerColor = Color.Transparent,
                unfocusedContainerColor = Color.Transparent)
        )
        Button(
            onClick = {
                if (siteName.isNotEmpty() && login.isNotEmpty() &&
password.isNotEmpty() && userID != null){
                    onAddEntry(PasswordEntry(siteName = siteName, login =
login, password = password, userId = userID))
                    siteName = ""
                    login = ""
                    password = ""
```

```
            }
        },
        modifier = Modifier.align(Alignment.End)
    ) {
        Text("Додати")
    }
  }
}
```

PasswordViewModel

```
class PasswordViewModel (application: Application):
AndroidViewModel(application){
  private val passwordDao =
AppDatabase.getDatabase(application).passwordDao()
  private val userDao = AppDatabase.getDatabase(application).userDao()
  private val _allEntries = MutableLiveData<List<PasswordEntry>>()
  val allEntries: LiveData<List<PasswordEntry>> = _allEntries
  private val _currentUser = MutableLiveData<User?>()
  val currentUser: LiveData<User?> = _currentUser

  private val cpAbeEncryption = CPABEEncryption()
  private val geoLocationManager = GeoLocationManager(application)

  fun insert(entry: PasswordEntry, currentUser: User) {
    // Шифруємо пароль
    val attributes = if (currentUser.isAdmin) {
      setOf("isAdmin", "geolocation")
    } else {
```

```
            setOf("geolocation")
        }

        val encryptedPassword = cpAbeEncryption.encrypt(entry.password,
attributes, pairing.getG2().newRandomElement())
        entry.password = encryptedPassword

        viewModelScope.launch {
            passwordDao.insert(entry)
        }
    }
    fun getCurrentUser(username: String, password: String): User? {
        val user = userDao.authenticateUser(username, password)
        return if (user != null) {
            // Перевірка геолокації при авторизації
            val currentLocation = getCurrentLocation()
            if (geoLocationManager.isLocationChanged(currentLocation)) {
                // Заборонити авторизацію, якщо геолокація змінилася
                null
            } else {
                geoLocationManager.saveLocation(currentLocation)
                user
            }
        } else {
            null
        }
    }

    fun delete(entry: PasswordEntry) = viewModelScope.launch {
        passwordDao.delete(entry)
```

```
    }
    fun getEntriesForUser(userId: Int){
        viewModelScope.launch {
            _allEntries.postValue(passwordDao.getEntriesForUserLive(userId))
        }
    }
    private fun getCurrentLocation(): Location {
        // Отримуємо поточну геолокацію користувача
        val location = Location("currentLocation")
        location.latitude = 50.4501
        location.longitude = 30.5236
        return location
    }
}
}
```

Cypher.kt

```
import org.openssl.pairing.Element
import org.openssl.pairing.Pairing
import org.openssl.pairing.PairingFactory
import org.openssl.pairing.pairing
import java.util.*

class CPABEEncryption {
    private val pairing: Pairing =
PairingFactory.getPairing("params/a.properties")

    // Структура, яка містить відкриті й закриті ключі
```

```
    data class KeyPair(
        val publicKey: Element, // Публічний ключ
        val masterKey: Element  // Майстер-ключ
    )

    // Структура для шифрованого повідомлення
    data class EncryptedData(
        val ciphertext: Element, // Зашифровані дані
        val accessPolicy: String // Політика доступу
    )

    // Генерація ключів
    fun setup(): KeyPair {
        val g = pairing.g1.newRandomElement().getImmutable()
        val alpha = pairing.zr.newRandomElement().getImmutable()
        val publicKey = g.powZn(alpha)
        val masterKey = alpha
        return KeyPair(publicKey, masterKey)
    }

    // Генерація закритого ключа для атрибутів
    fun generatePrivateKey(attributes: Set<String>, masterKey: Element):
Map<String, Element> {
        val privateKey = mutableMapOf<String, Element>()
        attributes.forEach { attribute ->
            val hashAttribute =
pairing.getZr().newElementFromHash(attribute.toByteArray(), 0,
attribute.length)
            privateKey[attribute] =
hashAttribute.powZn(masterKey).getImmutable()
```

```
        }
        return privateKey
    }

    // Шифрування даних
    fun encrypt(data: String, accessPolicy: String, publicKey: Element):
EncryptedData {
        val message =
pairing.getGT().newElementFromBytes(data.toByteArray()).getImmutable()
        val randomKey = pairing.zr.newRandomElement().getImmutable()

        // Зашифровані дані
        val ciphertext = message.mul(publicKey.powZn(randomKey))
        return EncryptedData(ciphertext, accessPolicy)
    }

    // Дешифрування даних
    fun decrypt(encryptedData: EncryptedData, privateKey: Map<String,
Element>, attributes: Set<String>): String? {
        val policyAttributes = encryptedData.accessPolicy.split(",").toSet()
        if (!policyAttributes.all { it in attributes }) {
            return null // Політика доступу не виконана
        }

        // Відновлення даних
        var decryptionKey =
pairing.getGT().newOneElement().getImmutable()
        attributes.forEach { attribute ->
            val keyPart = privateKey[attribute]
            if (keyPart != null) {
```

```
            decryptionKey = decryptionKey.mul(keyPart)
        }
    }

    // Відновлення тексту
    val decrypted =
encryptedData.ciphertext.div(decryptionKey).toBytes()
    return String(decrypted)
  }
}
```

DAO.kt

```
package com.example.passwordmanager

import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import androidx.room.Update

@Dao
interface PasswordDao{
  @Insert(onConflict = OnConflictStrategy.REPLACE)
  suspend fun insert(entry: PasswordEntry)

  @Update
```

```
    suspend fun update(entry: PasswordEntry)

    @Delete
    suspend fun delete(entry: PasswordEntry)

    @Query("SELECT * FROM password_entries ORDER BY siteName
ASC")
    fun getAllEntriesLive(): LiveData<List<PasswordEntry>>

    @Query("SELECT * FROM password_entries WHERE userId =
:userId")
    suspend fun getEntriesForUserLive(userId: Int): List<PasswordEntry>
}

@Dao
interface UserDao {
    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertUser(user: User)

    @Query("SELECT * FROM users WHERE username = :username AND
password = :password")
    suspend fun authenticateUser(username: String, password: String): User?

    @Query("SELECT * FROM users")
    suspend fun getAllUsers(): List<User>?
}
```

DB.kt

```
package com.example.passwordmanager

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import androidx.room.migration.Migration
import androidx.sqlite.db.SupportSQLiteDatabase

@Database(entities = [PasswordEntry::class, User::class], version = 2)
abstract class AppDatabase : RoomDatabase(){
    abstract fun passwordDao(): PasswordDao
    abstract fun userDao(): UserDao

    companion object{
        @Volatile private var instance: AppDatabase? = null
        private val MIGRATION_1_2 = object : Migration(1, 2) {
            override fun migrate(database: SupportSQLiteDatabase) {
                database.execSQL("ALTER TABLE password_entries ADD
COLUMN userId INTEGER NOT NULL DEFAULT 0")
                database.execSQL("""
            CREATE TABLE IF NOT EXISTS users (
                userId INTEGER PRIMARY KEY AUTOINCREMENT NOT
NULL,
                username TEXT NOT NULL,
                password TEXT NOT NULL,
                isAdmin INTEGER NOT NULL DEFAULT 0
            )
        """)
            }
```

```
        }

        fun getDatabase(context: Context): AppDatabase =
            instance ?: synchronized(this){
                instance ?: Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "password database"
                ).addMigrations(MIGRATION_1_2)  // Додаємо міграцію, якщо необхідно
                    .build().also { instance = it }
            }
    }
}
```