Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: "Аналіз застосування криптографічних методів для побудови системи управління паролями"

Виконав: студент

Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)

Слободян П.П.
підпис (прізвище та ініціали)

Керівник Деркач М.В.
підпис (прізвище та ініціали)

Нормоконтроль Стадник М.А.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Слободян Поліні Петрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз застосування криптографічних методів для побудови системи управління паролями

Керівник роботи Деркач Марина Володимирівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 24 » 11 2025 року № 4/7-1024

2. Термін подання студентом завершеної роботи 18.12.2025

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Галузь застосування систем управління паролями та їх значення для забезпечення інформаційної безпеки
Класифікація та аналіз сучасних менеджерів паролів
Основні загрози безпеці, пов'язані з використанням паролів
Криптографічні алгоритми шифрування, хешування та виведення ключів для захисту облікових даних
Математичні основи та принципи роботи сучасних криптографічних механізмів захисту паролів
Принципи End-to-End Encryption та Zero-Knowledge у менеджерах паролів
Механізми автентифікації та контролю доступу до зашифрованих сховищ
Розроблення, реалізація та тестування прототипу менеджера паролів
Питання безпеки життєдіяльності та основи охорони праці та висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Вступ, Актуальність теми, Мета та завдання дослідження, Об'єкт і предмет дослідження, Аналіз існуючих систем управління паролями, Криптографічні алгоритми та принципи безпеки, Архітектура розробленої системи, Реалізація програмного прототипу, Тестування та оцінка безпеки, Результати дослідження, Практичне значення та впровадження, Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., зав. каф. КС		
Безпека в надзвичайних ситуаціях	Теслюк В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 19.09.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	10.09 – 17.09	Виконано
2.	Огляд та аналіз наукових джерел з проблем управління паролями та інформаційної безпеки	18.09 – 30.09	Виконано
3.	Аналіз актуальності проблеми та сучасного стану досліджень у галузі управління паролями	01.10 – 07.10	Виконано
4.	Огляд та класифікація сучасних менеджерів паролів	08.10 – 14.10	Виконано
5.	Аналіз основних загроз безпеці та ризиків, пов’язаних з використанням паролів	15.10 – 21.10	Виконано
6.	Узагальнення результатів теоретичного аналізу та формування висновків першого розділу	22.10 – 25.10	Виконано
7.	Дослідження криптографічних алгоритмів шифрування та захисту облікових даних	26.10 – 02.11	Виконано
8.	Аналіз математичних основ алгоритмів шифрування, хешування та виведення ключів	03.11 – 09.11	Виконано
9.	Проєктування та реалізація прототипу програмної системи	10.11 – 24.11	Виконано
10.	Тестування функціональності та безпеки розробленої системи	25.11 – 05.12	Виконано
11.	Аналіз питань безпеки життєдіяльності та охорони праці під час роботи з програмними системами	06.12 – 10.12	Виконано
12.	Оформлення кваліфікаційної роботи, проходження нормоконтролю та перевірки на плагіат	11.12 – 16.12	Виконано
13.	Попередній захист кваліфікаційної роботи	17.12 – 18.12	Виконано
14.	Захист кваліфікаційної роботи	23.12.2025	

Студент _____ Слободян П.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Деркач М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз застосування криптографічних методів для побудови системи управління паролями // Кваліфікаційна робота ОР «Магістр» // Слободян Поліна Петрівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки // Тернопіль, 2025 // С. 81, рис. – 14, табл. – 5, крес. – 12, додат. – 2.

Ключові слова: менеджер паролів, інформаційна безпека, криптографія, шифрування, автентифікація, захист даних.

Кваліфікаційна робота присвячена аналізу сучасних підходів до захисту облікових даних користувачів та розробленню прототипу менеджера паролів із використанням сучасних криптографічних алгоритмів і механізмів безпеки. У роботі розглянуто актуальні загрози, пов'язані з використанням паролів, та обґрунтовано доцільність застосування спеціалізованих систем управління паролями для зниження ризиків несанкціонованого доступу.

У першому розділі кваліфікаційної роботи проаналізовано галузі застосування менеджерів паролів, розглянуто класифікацію сучасних рішень та виконано огляд наукових і технічних джерел, що стосуються проблем безпечного зберігання та використання паролів. У другому розділі кваліфікаційної роботи досліджено криптографічні механізми захисту облікових даних, зокрема алгоритми шифрування, хешування та виведення ключів. Розглянуто математичні основи алгоритмів AES, XChaCha20-Poly1305, PBKDF2, bcrypt, scrypt та Argon2id, а також принципи End-to-End Encryption і Zero-Knowledge. У третьому розділі кваліфікаційної роботи реалізовано прототип менеджера паролів із графічним інтерфейсом користувача, розроблено механізми автентифікації, шифрування та захисту збережених даних, а також проведено тестування функціональності та безпеки розробленої системи.

ABSTRACT

Analysis of the Application of Cryptographic Methods for Building a Password Management System // Qualification Thesis for the Degree of Master // Slobodian Polina Petrivna // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity // Ternopil, 2025 // p. 81, figs. 14, tbls 5, drws 12, apps – 2.

Keywords: password manager, information security, cryptography, encryption, authentication, data protection.

The qualification thesis is devoted to the analysis of modern approaches to protecting user credentials and to the development of a password manager prototype using contemporary cryptographic algorithms and security mechanisms. The paper considers current threats associated with password usage and substantiates the feasibility of applying specialized password management systems to reduce the risks of unauthorized access.

The first chapter analyzes the application areas of password managers, presents a classification of modern solutions, and reviews scientific and technical sources related to the problems of secure password storage and usage. The second chapter investigates cryptographic mechanisms for protecting credentials, including encryption, hashing, and key derivation algorithms. The mathematical foundations of the AES, XChaCha20-Poly1305, PBKDF2, bcrypt, scrypt, and Argon2id algorithms are examined, as well as the principles of End-to-End Encryption and Zero-Knowledge. The third chapter presents the implementation of a password manager prototype with a graphical user interface, describes the developed authentication, encryption, and data protection mechanisms, and provides the results of functional and security testing of the developed system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AES	–	симетричний алгоритм шифрування
AES-GCM	–	режим AES із перевіркою цілісності даних
Argon2id	–	алгоритм захисту паролів
E2EE	–	наскрізне шифрування даних
GUI	–	графічний інтерфейс
Hash-функція	–	перетворення даних у хеш
ISMS	–	система управління інформаційною безпекою
ISO/IEC 27001	–	стандарт інформаційної безпеки
JSON	–	формат зберігання даних
KDF	–	функція виведення ключа
Nonce	–	одноразове випадкове значення
PBKDF2	–	алгоритм формування ключа з пароля
Salt	–	випадкова добавка до пароля
XChaCha20-Poly1305	–	алгоритм автентифікованого шифрування
Zero-Knowledge	–	система не зберігає паролі

ВСТУП

Актуальність теми. У сучасному цифровому світі питання інформаційної безпеки є особливо актуальним. Зростання кількості онлайн-сервісів, мобільних додатків та корпоративних систем означає, що користувачі щодня отримують доступ до десятків ресурсів, кожен з яких потребує автентифікації. За таких обставин система управління паролями стає ключовим елементом інформаційної безпеки, оскільки від її надійності залежить безпека конфіденційних даних користувачів та організацій.

Паролі залишаються одним із найпоширеніших методів автентифікації, незважаючи на розвиток біометрії та багатофакторної автентифікації. Водночас неправильне зберігання або передача паролів часто призводить до витоку даних. Тому використання криптографічних методів, таких як хешування, шифрування та генерація криптографічно-стійких ключів, є важливим для створення безпечної системи управління автентифікацією.

Розробка ефективної системи управління паролями вимагає ретельного аналізу існуючих криптографічних алгоритмів, оцінки їхньої стійкості до атак та вибору оптимальних методів для конкретних завдань зберігання, передачі та оновлення паролів. Крім того, увага приділяється організації архітектури програмного забезпечення для забезпечення безпечної взаємодії користувача з системою без компрометації конфіденційних даних.

Мета цієї кваліфікаційної роботи магістра - є розроблення комплексного підходу до побудови безпечної системи управління паролями, що базується на використанні сучасних криптографічних алгоритмів, принципів безпечного зберігання інформації та механізмів аналізу надійності автентифікаційних даних.

Завданням дослідження є побудова концептуальної моделі системи управління паролями, яка поєднувала б сучасні криптографічні підходи, механізми захисту від витоків та зручність використання для кінцевого користувача. Така модель має ґрунтуватися на Zero-Knowledge принципах, сучасних алгоритмах хешування та шифрування, а також передбачати

комплексний аналіз надійності паролів, включаючи перевірку на наявність у витоках та оцінювання їхньої ентропії.

Об’єктом дослідження в цій роботі є системи управління паролями, що забезпечують зберігання, обробку та захист конфіденційних автентифікаційних даних користувачів.

Предметом дослідження є криптографічні методи, механізми захисту та моделі безпечної взаємодії користувача з системами управління паролями, зокрема алгоритми шифрування, хешування, генерування ключів та механізми оцінювання надійності паролів.

Наукова новизна цієї роботи полягає у розробці концептуальної моделі системи управління автентифікацією, яка поєднує сучасні підходи до хешування, шифрування та генерації ключів.

Практичне значення одержаних результатів полягає в потенційному застосуванні розроблених рішень для підвищення інформаційної безпеки в корпоративних та персональних системах управління паролями.

Апробація результатів магістерської роботи відбулась на XIII науково-технічній конференції «Інформаційні моделі, системи та технології» (м. Тернопіль, 17–18 грудня 2025 р.) у вигляді тез доповіді.

Публікації. Слободян П., Богач М., Деркач М. Аналіз популярних систем управління паролями // Матеріали XIII науково-технічної конференції «Інформаційні моделі, системи та технології». – Тернопіль, 2025. – С. 86.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ СИСТЕМ УПРАВЛІННЯ ПАРОЛЯМИ

1.1 Проблематика зберігання та управління паролями у цифровому середовищі

У сучасному світі цифрових технологій безпечне управління паролями є одним із ключових аспектів забезпечення інформаційної безпеки як для окремих користувачів, так і для організацій. Паролі залишаються основним і найпоширенішим механізмом автентифікації, який застосовується у більшості сервісів та програмних продуктів. Вони виконують функцію захисного бар'єру, який дозволяє відокремлювати доступ до особистої, конфіденційної та корпоративної інформації, забезпечуючи контроль над тим, хто може отримати доступ до системи та її даних [1].

Проте у практичній сфері управління паролями існує ряд проблем, які суттєво ускладнюють забезпечення надійного захисту даних. По-перше, користувачі часто створюють занадто прості або повторювані паролі, щоб полегшити їх запам'ятовування. Це відбувається через постійну необхідність одночасного використання великої кількості облікових записів у різних сервісах, що створює психологічне навантаження, відоме як «парольна втома» (password fatigue) [2]. Дослідження Beyond Identity показує, що така втома значно знижує обачність користувачів і підвищує ймовірність компрометації їх облікових записів, навіть якщо вони використовують менеджери паролів [3].

По-друге, без належної організації процесу управління паролями виникає ризик ненадійного зберігання облікових даних. Деякі користувачі зберігають паролі у відкритих текстових файлах, нотатках або в браузерах без додаткового шифрування, що створює пряму загрозу для безпеки їхніх даних. Крім того, у багатьох випадках паролі передаються через незахищені канали зв'язку або використовуються без регулярної зміни, що також сприяє підвищенню ризику несанкціонованого доступу [4]. У медичних і корпоративних установах також фіксуються випадки неправильного зберігання або спільного використання паролів. Наприклад, HIPAA Journal повідомляє про численні інциденти, коли

працівники залишали пароль у вигляді текстових нотаток або передавали їх через незахищені канали [14].

По-третє, існуючі системи управління паролями не завжди забезпечують баланс між безпекою та зручністю використання. Багато користувачів уникають складних процедур автентифікації або багатофакторної перевірки через складність і тривалість процесу входу, що підвищує ймовірність використання слабких або повторюваних паролів. Це створює ситуацію, коли технічні засоби захисту існують, але через людський фактор вони використовуються неефективно, що підкреслює необхідність комплексного підходу до проблеми [1, 3].

У звітах HIPAA Journal наведені показові інциденти, пов’язані з неправильним використанням менеджерів паролів. Наприклад: користувачі зберігали майстер-пароль у файлі passwords.txt на робочому столі; корпоративні паролі зберігалися у спільному Google Docs без обмеження доступу; паролі до критичних медичних систем передавались через електронну пошту без шифрування. У більшості хмарних менеджерів (NordPass, Bitwarden) при введенні слабкого або скомпрометованого пароля користувач бачить попередження: “This password has appeared in a known data breach” (див. рис. 1.1). Це демонструє реальність загрози та масштаби проблеми.

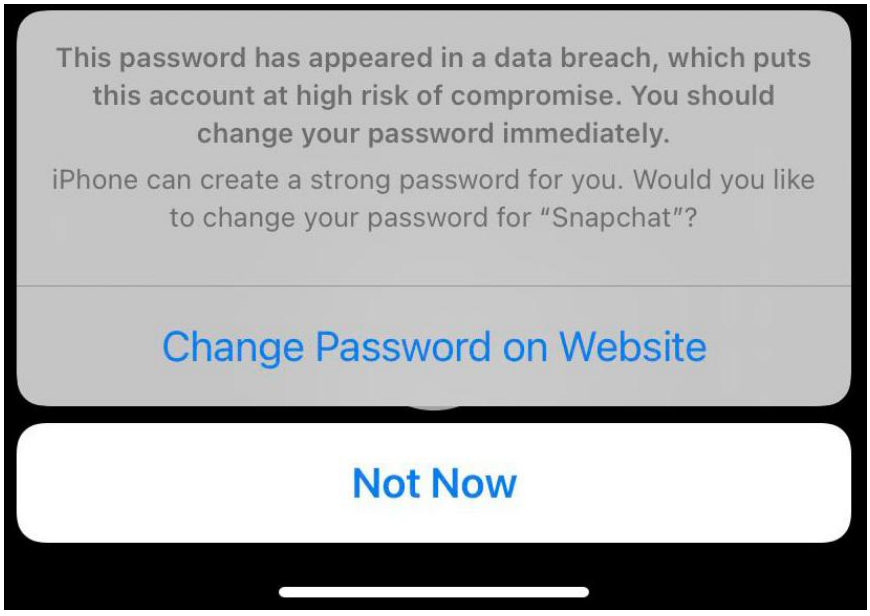

Рисунок 1.1 – Попередження про знайдений у витоках пароль

Механізм роботи таких попереджень базується на використанні сервісу HaveIBeenPwned, який застосовує модель конфіденційності k-Anonymity, що дозволяє перевірити пароль, не передаючи його у відкритому вигляді [16].

Крім індивідуальних користувачів, проблема зберігання паролів є надзвичайно актуальною для організацій. У корпоративному середовищі велика кількість співробітників потребує одночасного доступу до різних систем і баз даних, що значно ускладнює контроль за обліковими даними. Невірна організація доступів, відсутність політик регулярної зміни паролів та недостатній рівень шифрування можуть призводити до витоків конфіденційної інформації, фінансових збитків і втрати довіри клієнтів або партнерів [1, 4].

Попри широке впровадження сучасних криптографічних стандартів у системах управління паролями, реальний рівень їхньої стійкості визначається не лише силою алгоритмів, а насамперед якістю їхньої реалізації. На практиці значна частина вразливостей виникає не через недоліки в математичній основі алгоритмів, а через неправильну конфігурацію, спрощення параметрів, відсутність належного контролю за генерацією ключів або використання застарілих технологій. Саме тому менеджери паролів, навіть базовані на перевірених криптографічних принципах, можуть залишатися вразливими, якщо розробники або користувачі нехтують сучасними рекомендаціями щодо безпеки.

Однією з найсерйозніших проблем є застосування швидких або застарілих хеш-функцій під час захисту майстер-пароля. Алгоритми, такі як SHA-1 чи SHA-256, взагалі є безпечними як хеш-функції, але в контексті захисту паролів їхня висока швидкість робить перебір надзвичайно ефективним на сучасних графічних процесорах. Саме тому міжнародні рекомендації, зокрема NIST SP 800-63B, наполягають на використанні пам’яттєво-складних алгоритмів хешування, таких як Argon2id або scrypt, які забезпечують значно вищу стійкість до перебору за рахунок великих обчислювальних витрат [28]. Відсутність таких алгоритмів або використання їх з мінімальними параметрами суттєво знижує рівень захисту всієї системи.

Ще одним важливим ризиком є некоректне застосування механізмів шифрування, що може призвести до компрометації даних навіть попри використання формально безпечних алгоритмів. Наприклад, у деяких реалізаціях AES розробники неправильно налаштовують режими шифрування або використовують передбачувані ініціалізаційні вектори, що полегшує аналіз структури зашифрованих даних. Подібні ситуації неодноразово згадуються у фаховій літературі з прикладної криптографії, яка наголошує, що людські помилки та спрощення реалізації становлять більшу загрозу, ніж математичні атаки на алгоритми [21].

Серйозним фактором ризику є також використання неякісних генераторів випадкових чисел, особливо під час створення ключів або сольових значень. Якщо генератор не має достатньої ентропії або є передбачуваним, то навіть стійкі криптографічні алгоритми втрачають свій захист. Це може проявлятись у повторенні сольових значень, слабких ключах шифрування або можливості вгадування параметрів, що використовуються для захисту сховища. Не випадково у криптографічних стандартах наголошується на необхідності використання криптографічно безпечних генераторів випадкових чисел (CSPRNG) для будь-яких операцій, пов’язаних із ключами.

Іншою суттєвою проблемою залишається некоректне впровадження багатофакторної автентифікації. Якщо MFA налаштована поверхнево або прив’язана до нешифрованих каналів (наприклад, SMS-кодів), це створює ілюзію безпеки, тоді як справжній рівень захисту залишається низьким. Саме тому сучасні стандарти автентифікації, такі як WebAuthn і FIDO2, рекомендують використовувати апаратні ключі або криптографічні протоколи з доведеною стійкістю до фішингових атак [33]. Відсутність таких механізмів або їх неправильна інтеграція роблять систему вразливою навіть за наявності сильного шифрування.

Окрему категорію вразливостей становлять помилки під час реалізації перевірки паролів на витоки. Використання API сервісів перевірки без реалізації моделей приватності може створювати додаткові ризики витоку інформації про

користувача або його паролі. Тому системи, які перевіряють хеші паролів за повним значенням або передають небезпечні параметри запиту, порушують принципи конфіденційності, які художньо вирішені у моделі k-Anonymity, реалізованій у сервісі HaveIBeenPwned [36].

Значної шкоди безпеці також можуть завдати помилки у зберіганні допоміжних секретів, таких як “pepper”. Якщо цей секрет зберігається у тому ж сховищі, що й паролі, або не належним чином захищений, ефективність додаткового шару безпеки зводиться нанівець. У наукових роботах звертається увага на те, що поділ секретів, їх розподілене зберігання та ізоляція – ключові принципи захисту систем, що мають критично важливі дані [21]. Нарешті, серйозною практичною загрозою є інтеграційні вразливості, які виникають у менеджерах паролів через взаємодію з браузерами, операційними системами або сторонніми розширеннями. Ін’єкції скриптів, атаки через автозаповнення форм або перехоплення буфера обміну можуть стати причиною витоку паролів, навіть якщо криптографічна база системи є надійною. Таким чином, криптографія забезпечує математичну стійкість, але не гарантує загальної безпеки у разі помилок на рівні інтеграції.

Таким чином, проблематика зберігання та управління паролями охоплює як технічні аспекти, так і поведінкові та психологічні чинники. Надійне управління паролями передбачає застосування сучасних криптографічних алгоритмів для безпечного зберігання та передачі облікових даних, використання багатофакторної автентифікації, а також розробку правил і політик безпечного користування системами. Проте більшість ризиків, пов'язаних із менеджерами паролів, виникають через неправильно налаштовані або спрощені реалізації, а не через слабкість сучасних криптографічних алгоритмів. Невірно підібрані параметри хешування, використання застарілих алгоритмів, слабкі генератори випадкових чисел, вразливості у каналах передачі або поверхнева інтеграція багатофакторної автентифікації можуть повністю нівелювати криптографічний захист.

З огляду на це, дослідження сучасних рішень у сфері управління паролями та аналіз проблем, пов’язаних із їх використанням, є надзвичайно актуальними. Вирішення цих проблем дозволяє забезпечити надійний захист даних, підвищити ефективність роботи користувачів та зміцнити загальний рівень інформаційної безпеки в цифровому середовищі. Саме тому забезпечення належної реалізації криптографічних механізмів є критично важливим етапом проєктування будь-якої системи управління паролями.

1.2 Класифікація систем управління паролями

Системи управління паролями є важливою складовою сучасних інформаційних систем і спрямовані на забезпечення безпечного зберігання та обробки облікових даних користувачів. У зв’язку з постійним збільшенням кількості цифрових сервісів та платформ, необхідність ефективного управління паролями стає все більш актуальною. Системи управління паролями дозволяють користувачам зберігати, організовувати та використовувати складні паролі, не ризикуючи їхньою безпекою, і тим самим знижують ймовірність компрометації облікових записів [1].

Залежно від архітектури та способу зберігання даних, сучасні системи управління паролями можна умовно поділити на кілька основних категорій: локальні, хмарні та корпоративні.

Локальні системи управління паролями передбачають зберігання всіх облікових даних безпосередньо на пристрої користувача. Такі рішення зазвичай надають високий рівень конфіденційності, оскільки доступ до паролів має лише власник пристрою. Водночас їхній недолік полягає у відсутності автоматичної синхронізації між різними пристроями, а також у високій залежності від безпеки самого пристрою. У випадку його пошкодження або втрати даних користувач ризикує остаточно втратити доступ до своїх облікових записів [1]. Прикладом такої системи є KeePass.

Переваги локальних рішень:

– повний контроль над даними, оскільки паролі не покидають пристрій;

– високий рівень криптографічного захисту (AES-256, Twofish);

– відсутність необхідності у хмарних серверах, що знижує ризики стороннього доступу.

Недоліки:

– відсутність автоматичної синхронізації між пристроями;

– ризик втрати даних у випадку пошкодження або втрати пристрою;

– складність налаштування для недосвідчених користувачів.

Дослідження зауважують, що локальні менеджери підходять переважно для технічно досвідчених користувачів, а також для організацій з високим рівнем вимог до локального контролю безпеки [14].

Хмарні системи управління паролями зберігають облікові дані на віддалених серверах, що дозволяє синхронізувати їх між кількома пристроями користувача, включаючи мобільні телефони, планшети та комп'ютери. Такі рішення надають високу зручність використання і забезпечують доступ до паролів у будь-який час і з будь-якого місця. Однак безпека цих систем повністю залежить від надійності серверної інфраструктури та застосованих криптографічних методів. Найпоширенішими прикладами таких систем є Bitwarden**,** NordPass**,** 1Password та LastPass**.** Хмарні менеджери часто реалізують механізми шифрування та багатоетапну автентифікацію, щоб мінімізувати ризики компрометації даних [1, 5].

Основні переваги:

– зручність – доступ із будь-якого пристрою;

– автоматична синхронізація;

– функції перевірки на витоки, моніторинг слабких і повторюваних паролів;

– Zero-Knowledge архітектура – провайдер не має доступу до даних користувача.

Недоліки:

– залежність від безпеки хмарної інфраструктури;
– потенційний ризик масових витоків у разі атаки на сервіс.

За даними NIST, хмарні менеджери можуть забезпечувати найвищий рівень захисту за умови належної криптографічної реалізації (Argon2, PBKDF2, XChaCha20, Zero-Knowledge) та активного багатофакторного захисту [17].

Корпоративні системи управління паролями призначені для організацій із великою кількістю користувачів. Вони поєднують централізоване управління обліковими даними, можливість контролю доступу, впровадження політик безпеки, автоматичне генерування складних паролів та інтеграцію з багатофакторною автентифікацією. Такі системи особливо важливі для компаній із підвищеними вимогами до конфіденційності інформації та високим ризиком кібератак. Вони дозволяють організувати контроль життєвого циклу пароля, включаючи його зміну, оновлення та блокування, а також забезпечують аудит активності користувачів для своєчасного виявлення потенційних загроз [1].

Корпоративні рішення використовуються у великих організаціях, де необхідний: централізований контроль доступу; журнали дій, аудит подій та моніторинг; інтеграція з Active Directory, LDAP, Azure AD; розмежування прав доступу; керування спільними обліковими записами. Такі системи використовують у ролі «централізованого сховища секретів» (Vault), забезпечуючи керування не лише паролями, а й токенами, ключами API, SSH-ключами тощо. NIST рекомендує корпоративним структурам впроваджувати політики: періодичного перегляду паролів; обов’язкової MFA; автоматизованого аналізу ризиків доступу [17]. До таких систем належать CyberArk, HashiCorp Vault, Keeper Enterprise, Thycotic Secret Server та BeyondTrust Password Safe. До переваг відносять те, що корпоративні менеджери паролів забезпечують:

– централізоване управління доступом до облікових даних;

– автоматичну зміну паролів і ведення журналів аудиту, що підвищує рівень безпеки та спрощує контроль доступу в організаціях.

Основними недоліками є:

– висока вартість впровадження;

– складність налаштування;

– залежність від централізованої інфраструктури, збій якої може тимчасово обмежити доступ до ресурсів.

У таблиці 1.1. представлено аналіз кожної категорії систем управління паролями, що наведені вище.

Таблиця 1.1 – Класифікація систем управління паролями

Тип системи	**Переваги**	**Недоліки**	**Цільова аудиторія**
Локальні	Макс. конфіденційність, відсутність залежності від сервера	Немає синхронізації, ризик втрати бази	Технічні користувачі
Хмарні	Зручність, доступ з різних пристроїв, моніторинг витоків	Ризики атак на хмару	Звичайні користувачі, малий бізнес
Корпоративні	Централізований контроль, інтеграція з AD, аудит	Висока вартість, складність впровадження	Бізнес, організації, IT-відділи

У кожній групі використовуються різні криптографічні засоби, до прикладу, у локальній: AES-256, Twofish, SHA-256 / SHA-512, PBKDF2-HMAC-SHA256. Хмарні системи використовують Argon2id, XChaCha20-Poly1305, Zero-Knowledge Encryption та TLS 1.3 для передачі даних. А корпоративні в свою чергу – Hardware Security Modules (HSM), управління ключами (KMS), ротація ключів та паролів і захищене сховище секретів (Vault).

Окрім поділу за архітектурою, системи управління паролями можна класифікувати за рівнем інтеграції та функціональними можливостями. Деякі

менеджери орієнтовані на приватних користувачів і пропонують мінімальний набір функцій, достатній для повсякденного використання, тоді як інші рішення включають додаткові можливості, такі як спільне використання паролів, інтеграція з корпоративними каталогами, відстеження безпеки паролів та моніторинг витоків даних [5].

Наприклад, аналіз двох популярних рішень – NordPass та Bitwarden – демонструє, як архітектура впливає на можливості системи. NordPass реалізує хмарний підхід із централізованим зберіганням і синхронізацією, пропонуючи користувачам простий інтерфейс та сучасні методи шифрування. Bitwarden, навпаки, надає користувачу гнучкість: він підтримує як хмарний режим, так і можливість локального розгортання, що особливо важливо для користувачів, які прагнуть мати повний контроль над своїми даними. Це демонструє, що навіть у межах однієї категорії рішення можуть суттєво відрізнятися за рівнем безпеки, гнучкістю та зручністю використання [5].

Таким чином, класифікація систем управління паролями дозволяє не лише розрізняти технічні та функціональні особливості різних рішень, а й оцінювати їхню відповідність конкретним завданням користувачів та організацій. Вона допомагає зрозуміти, який тип системи буде найбільш ефективним у певному середовищі, і які заходи необхідно впроваджувати для забезпечення надійного захисту облікових даних, що є особливо актуальним у сучасних умовах цифрової безпеки [1, 5].

1.3 Огляд відомих рішень

Сучасний ринок систем управління паролями пропонує великий спектр рішень, призначених для різних категорій користувачів – від приватних осіб до великих корпоративних організацій. Основна мета таких систем полягає у забезпеченні надійного та зручного зберігання облікових даних, захисту від несанкціонованого доступу та спрощенні процесу користування великою кількістю облікових записів. Незважаючи на загальну спрямованість на

безпеку, менеджери паролів значно відрізняються між собою за архітектурою, функціональністю, способами синхронізації та рівнем захисту [6].

1.3.1 Bitwarden

Bitwarden (див рис. 1.2) є відкритим менеджером паролів, який підтримує хмарне зберігання даних та синхронізацію між пристроями. Його відкритий код дозволяє незалежним експертам проводити аудит безпеки та перевірку на наявність вразливостей, що підвищує довіру користувачів. Bitwarden пропонує широкий спектр функцій: автоматичне заповнення форм для входу на сайти, генерація складних паролів, спільний доступ до облікових даних у межах сім'ї або команди, а також інтеграцію з різними браузерами та мобільними платформами. Також він підтримує багатофакторну автентифікацію та шифрування даних за допомогою сучасних алгоритмів, що робить його одним із найбезпечніших і водночас зручних рішень для користувачів різного рівня підготовки [7, 6]. Усі паролі шифруються за допомогою AES-256, а ключ формується на основі PBKDF2-HMAC-SHA256, що відповідає сучасним вимогам до криптографічного захисту.

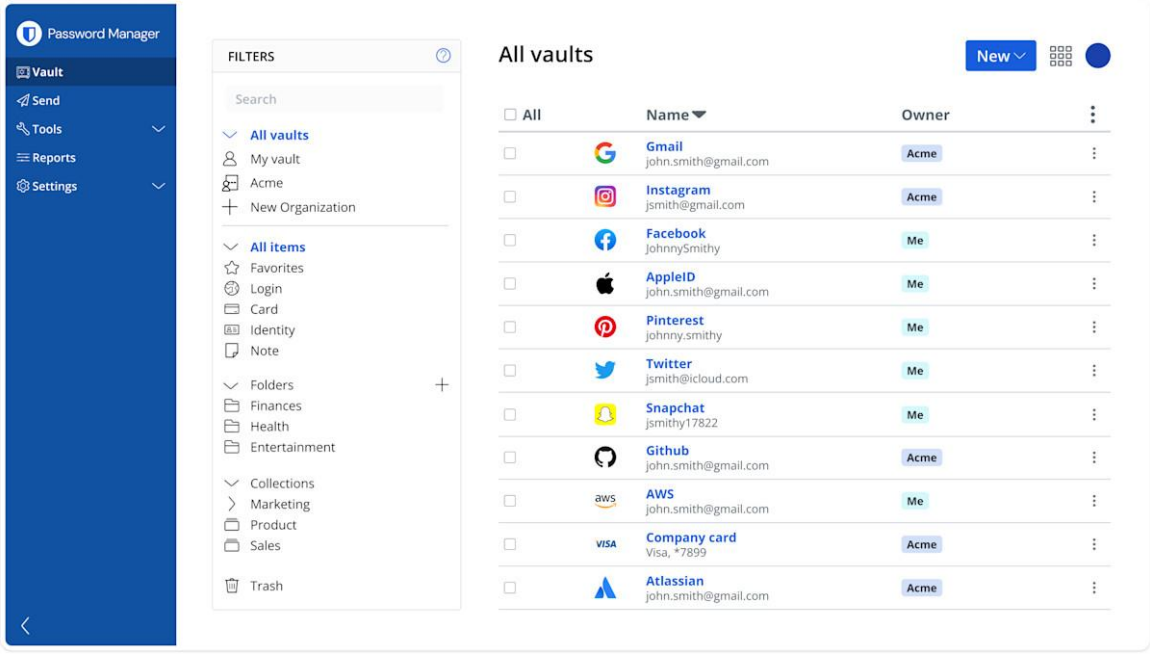

Рисунок 1.2 – Інтерфейс Bitwarden

Крім того, Bitwarden інтегрований із сервісом HaveIBeenPwned, що дозволяє виявляти скомпрометовані паролі та попереджати користувача про їх появу у базах витоків [20]. Завдяки поєднанню високого рівня безпеки та доступності Bitwarden є одним із найпоширеніших рішень серед приватних користувачів і малих команд.

1.3.2 KeePass

KeePass – це локальний менеджер паролів із відкритим вихідним кодом, який зберігає всі дані на пристрої користувача (див рис. 1.3). Його перевагою є повний контроль над інформацією: всі бази даних зашифровані за допомогою потужних алгоритмів, таких як AES або Twofish. Крім того, KeePass підтримує різноманітні плагіни для розширення функціоналу, наприклад, для автоматичного заповнення форм або інтеграції з браузерами. Головним недоліком локальних систем є відсутність автоматичної синхронізації між пристроями, що може створювати труднощі для користувачів із кількома девайсами. Проте для тих, хто прагне максимального контролю над власними даними та мінімальної залежності від сторонніх серверів, KeePass залишається оптимальним вибором [8, 6]. Його бази даних зберігаються на пристрої у зашифрованому вигляді та можуть захищатися як майстер-паролем, так і ключовим файлом. KeePass підтримує широкий набір криптографічних алгоритмів, зокрема AES-256, Twofish та ChaCha20, що дозволяє користувачеві самостійно обирати бажаний рівень захисту.

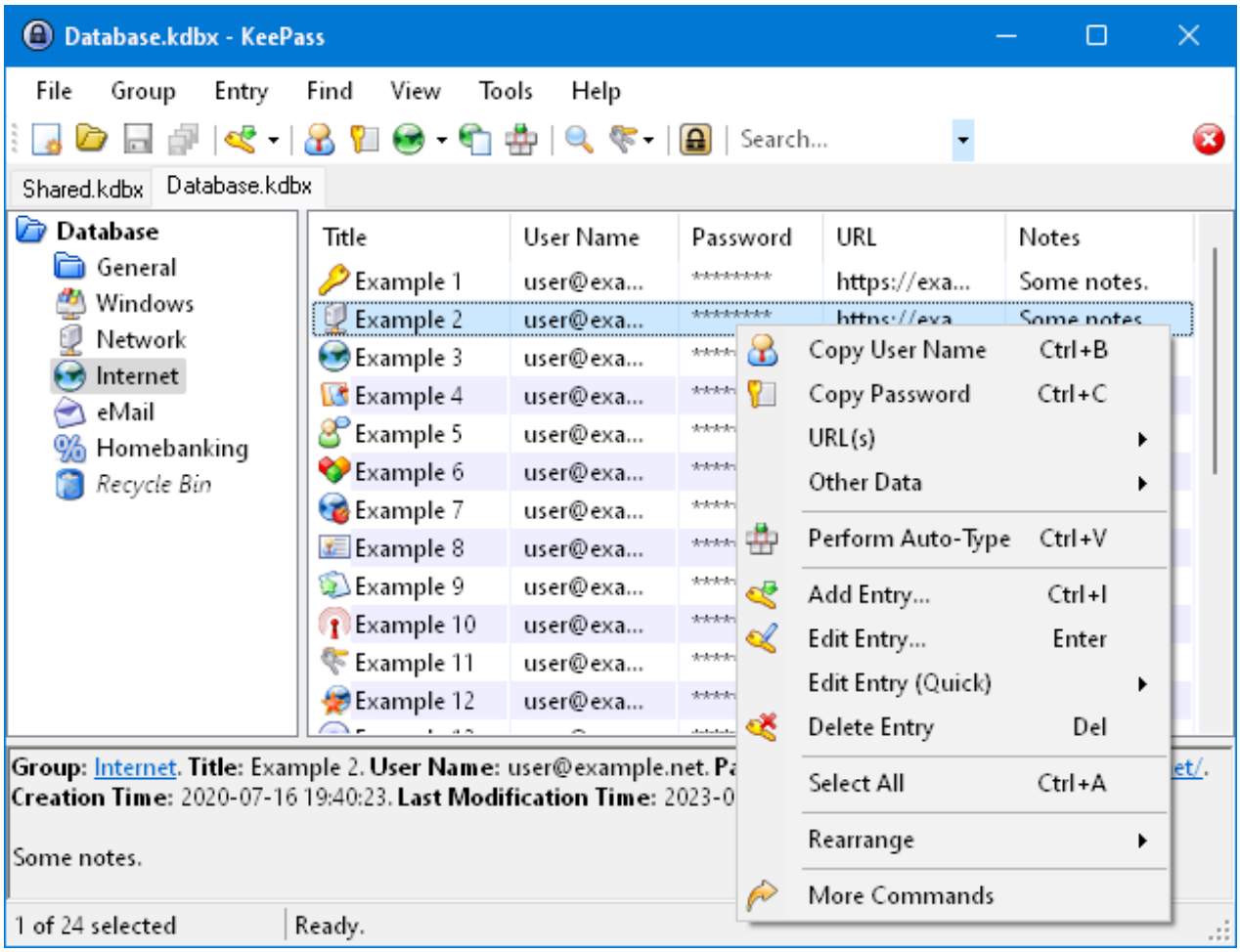

Рисунок 1.3 – Інтерфейс KeePass

Водночас KeePass потребує самостійного налаштування та може бути складнішим у використанні для недосвідчених користувачів. Він не пропонує автоматичної синхронізації, проте дозволяє додавати численні плагіни, які розширюють його можливості. Таким чином, KeePass підходить користувачам, які ставлять пріоритет на локальну безпеку та готові самостійно керувати конфігурацією системи [8].

1.3.3 1Password

1Password орієнтований на комерційних користувачів та корпоративні потреби, пропонуючи зручний інтерфейс та комплексний набір функцій безпеки (див рис. 1.4). Система забезпечує хмарну синхронізацію між пристроями, підтримку багатофакторної автентифікації, а також створення

окремих «сейфів» для груп користувачів або команд. 1Password дозволяє не лише зберігати паролі, а й інші конфіденційні дані, наприклад, ліцензійні ключі, документи або дані платіжних карток.

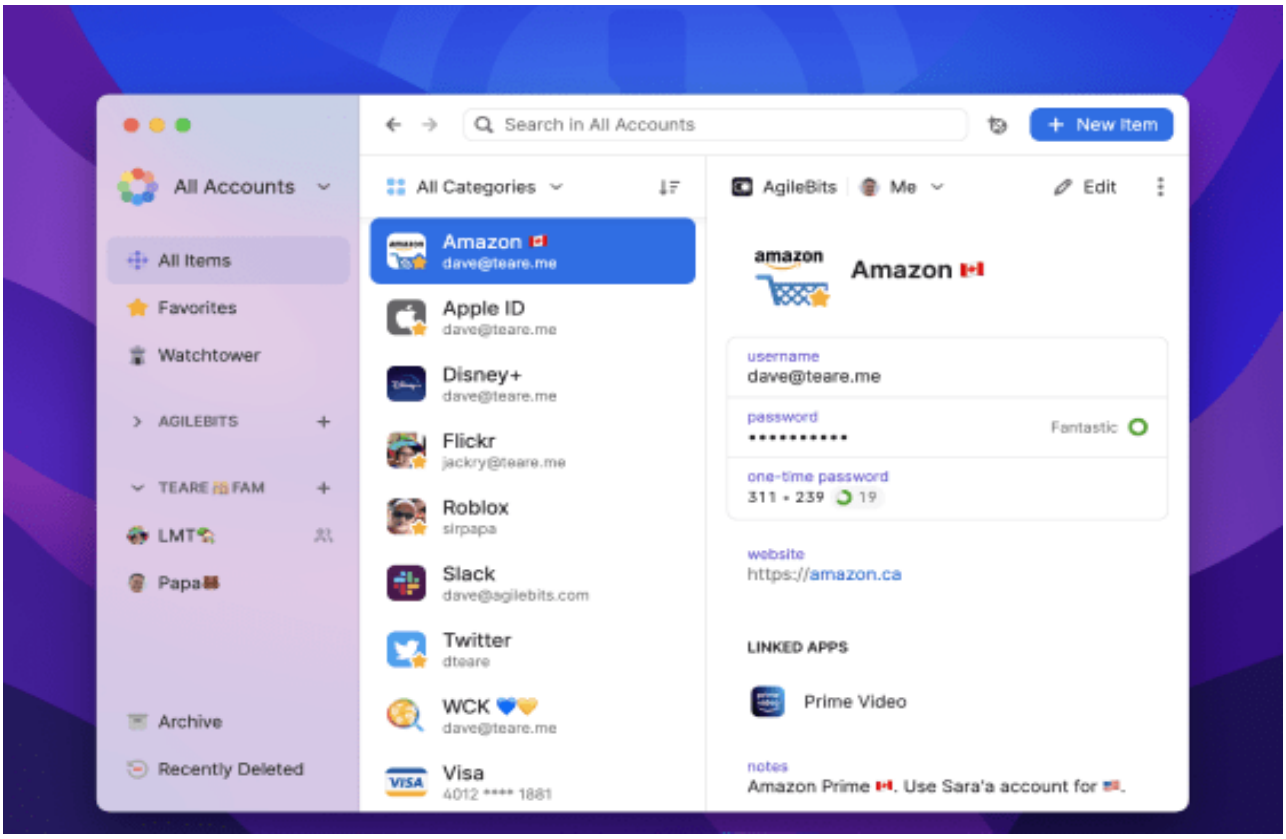

Рисунок 1.4 – Інтерфейс 1Password

Однією з ключових переваг є інтеграція з корпоративними каталогами та системами управління доступом, що робить його придатним для великих організацій, де важливо не лише зберігати паролі, а й контролювати, хто і коли отримує доступ до інформації [9, 6]. Для генерації ключів використовується алгоритм Argon2id, що відповідає сучасним рекомендаціям щодо захисту від перебору. Також 1Password має інтеграцію з корпоративними каталогами доступу, що дозволяє централізовано керувати обліковими записами, встановлювати політики безпеки та контролювати взаємодію персоналу з конфіденційними даними [9].

1.3.4 NordPass

NordPass є сучасним хмарним рішенням від компанії Nord Security (див рис. 1.5). Він пропонує зручний та простий інтерфейс, синхронізацію між пристроями, автоматичне заповнення форм та генерацію складних паролів. NordPass підтримує алгоритми шифрування XChaCha20 та Argon2, що забезпечує високий рівень безпеки даних користувачів. Сервіс також надає можливість зберігати не лише паролі, а й різні конфіденційні дані, наприклад, нотатки або інформацію про платіжні картки. Однією з цікавих функцій є перевірка на наявність слабких або скомпрометованих паролів, що дозволяє користувачам своєчасно оновлювати свої облікові дані та підвищувати загальний рівень безпеки [10, 6]. NordPass вирізняється простотою інтерфейсу, високою зручністю та сучасною криптографією. Він використовує алгоритм XChaCha20-Poly1305, який забезпечує високий рівень захисту при мінімальних затримках у роботі. Крім того, для захисту майстер-пароля застосовується Argon2id – алгоритм, який вважається одним із найстійкіших до атак перебором. NordPass пропонує інструменти для аналізу загального «здоров'я» сховища паролів, включаючи виявлення слабких, повторюваних або скомпрометованих паролів. Завдяки функціям моніторингу витоків, менеджер може автоматично повідомляти користувача про появу його даних у базах зламу, що значно підвищує рівень загальної кібербезпеки [10, 11, 13].

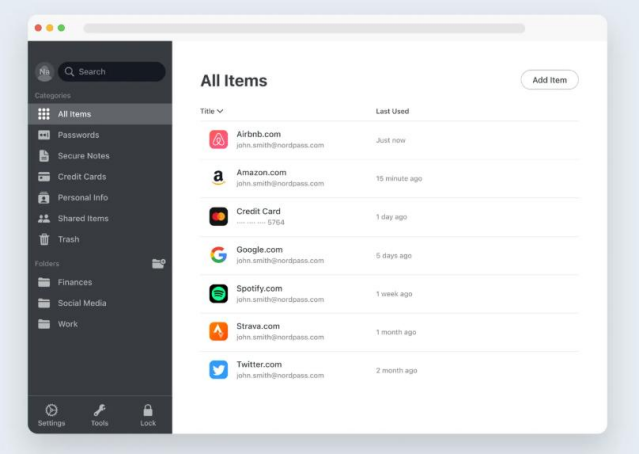

Рисунок 1.5 – Інтерфейс NordPass

Всебічний аналіз цих систем показує суттєві відмінності у підходах до організації даних, зручності використання та безпеки. Локальні менеджери, такі як KeePass, забезпечують максимальний контроль над інформацією, однак вимагають додаткових зусиль для резервного копіювання та синхронізації. Хмарні системи, такі як Bitwarden і NordPass, спрощують користування завдяки синхронізації на кількох пристроях і хмарному зберіганню, проте безпека даних значною мірою залежить від надійності серверної інфраструктури та застосованих криптографічних методів. 1Password поєднує зручність і безпеку, пропонуючи розширені корпоративні функції та високий рівень шифрування.

Крім основного функціоналу, сучасні менеджери паролів активно розвиваються і впроваджують додаткові інструменти для підвищення безпеки та зручності користувачів. Це включає інтеграцію з браузерами та мобільними додатками, автоматичне оновлення скомпрометованих паролів, відстеження витоків даних, спільний доступ у межах організації або сім'ї та можливість створення надійних паролів для різних типів акаунтів. Таким чином, сучасні менеджери паролів не обмежуються лише зберіганням паролів – вони стають комплексними платформами для управління цифровою безпекою та захисту персональних даних [6, 11, 12, 13].

Таблиця 1.2 – Аналіз популярних систем управління паролями

Характеристика	**Bitwarden**	**KeePass**	**1Password**	**NordPass**
Зберігання	Хмара / локально	Локально	Хмара	Хмара
Тип коду	Відкритий	Відкритий	Закритий	Закритий
Шифрування	AES-256	AES-256, Twofish	AES-256 + Secret Key	XChaCha20
Синхронізація	Так	Ні (вручну)	Так	Так
MFA	Так (TOTP, U2F)	Через плагіни	Так (широкий вибір)	Так
Перевірка витоків	Так	Через плагіни	Так	Так

Окрім основних можливостей, усі згадані системи активно впроваджують механізми аналізу складності паролів. Наприклад, Bitwarden, NordPass та 1Password використовують алгоритм zxcvbn, який аналізує структуру пароля, визначає логічні послідовності, словникові збіги й типові патерни введення. Це дає можливість оцінювати не лише довжину пароля, а й його реальну непередбачуваність та стійкість до підбору [15]. Хмарні менеджери також інтегрують API сервісу HaveIBeenPwned, що дозволяє порівнювати хеші паролів із базами вже відомих витоків, не розкриваючи їх у відкритому вигляді завдяки моделі k-Anonymity [16]. Таким чином, перевірка на наявність у витоках стала стандартом у сучасних менеджерах паролів і суттєво підвищує рівень захисту користувачів.

Підсумовуючи, можна сказати, що дійсно огляд популярних рішень показує, як сучасні менеджери паролів значно відрізняються між собою за функціональністю, підходом до архітектури, криптографічними алгоритмами та рівнем захисту. KeePass залишається оптимальним вибором для користувачів, оскільки надає максимальний рівень локального контролю та гнучкості, Bitwarden поєднує відкритий код із доступністю і високим рівнем безпеки, 1Password пропонує потужні корпоративні інструменти та додатковий рівень криптографічного захисту завдяки хмарній синхронізації, тоді як NordPass робить акцент на сучасних криптографічних алгоритмах, зручності та автоматизованому аналізі безпеки. Це різноманіття дозволяє користувачам обирати систему, яка найкраще відповідає їхнім технічним можливостям, потребам та вимогам до конфіденційності.

РОЗДІЛ 2 АНАЛІЗ КРИПТОГРАФІЧНОГО ЗАХИСТУ СИСТЕМ УПРАВЛІННЯ ПАРОЛЯМИ

2.1 Аналіз стандартів щодо зберігання облікових даних

Ефективність систем управління паролями визначається не лише якістю реалізації криптографічних алгоритмів, а й відповідністю міжнародним стандартам, які регламентують вимоги до автентифікації, зберігання паролів та організації політик доступу. Найвпливовішими у цій сфері є стандарти NIST SP 800-63B, рекомендації OWASP та норми ISO/IEC, які формують комплексний підхід до побудови захищених систем і визначають мінімальний рівень безпеки, необхідний для роботи з конфіденційною інформацією.

2.1.1 NIST SP 800-63B

Одним із базових документів є NIST SP 800-63B, який визначає сучасні вимоги до цифрової автентифікації. Документ наголошує, що паролі мають захищатися за допомогою пам’яттєво-складних алгоритмів хешування, таких як Argon2, PBKDF2 або scrypt, із великими параметрами витрат часу й пам’яті. NIST підкреслює важливість того, щоб системи не нав’язували надмірно складних вимог на кшталт примусових регулярних змін або обов’язкового використання спеціальних символів, оскільки такі практики знижують зручність користування та стимулюють людей до вибору передбачуваних патернів [28]. Стандарт також наголошує на необхідності перевірки паролів на наявність у відомих витоках, що стало сучасним підходом у хмарних менеджерах паролів.

2.1.2 OWASP ASVS

Важливу роль у формуванні вимог до безпеки відіграють рекомендації OWASP Application Security Verification Standard (ASVS) та супутні матеріали,

зокрема Password Storage Cheat Sheet. Ці документи деталізують вимоги до реалізації зберігання паролів: використовувати криптографічно безпечні генератори випадкових чисел, окреме зберігання “pepper”, унеможливлювати використання швидких хеш-функцій та забезпечувати захист каналів зв’язку через сучасні протоколи. OWASP значну увагу приділяє не лише криптографічній частині, а й загальній архітектурі системи, підкреслюючи важливість ізоляції секретів, багатофакторної автентифікації та запобігання витоку метаданих [32].

2.1.3 ISO/IEC 2700X

У сфері організації захисту інформації критично важливим є також міжнародний стандарт ISO/IEC 27001, який встановлює вимоги до систем управління інформаційною безпекою. Хоча він не визначає конкретних криптографічних алгоритмів, ISO/IEC 27001 регламентує необхідність створення політик доступу, управління автентифікаційними даними та мінімізації ризиків, пов’язаних із людським фактором.

У додатковому стандарті ISO/IEC 27002 детально описуються організаційні заходи, включаючи вимоги до створення складних паролів, регулярного аудиту доступів та сегментації відповідальності між адміністраторами систем. Таким чином, стандарти ISO задають загальну структуру управління безпекою, а NIST і OWASP деталізують технічні аспекти її реалізації. Із розвитком вебтехнологій важливим елементом систем управління паролями стали механізми сучасної автентифікації, які визначаються структурою стандартів FIDO2 та WebAuthn. Ці технології дозволяють використовувати апаратні ключі безпеки та криптографічні протоколи з доведеною стійкістю до фішингових атак, що значною мірою підсилює безпеку менеджерів паролів і хмарних сервісів. Згідно зі стандартами FIDO Alliance, ці механізми базуються на криптографії відкритих ключів і усувають потребу у традиційних паролях у процесі автентифікації, хоча не замінюють їх у внутрішніх сховищах даних [33].

Узагальнюючи, сучасні міжнародні стандарти та рекомендації формують єдину структуру вимог, яка поєднує технічні, організаційні та поведінкові аспекти безпеки автентифікаційних даних. NIST задає базові технічні критерії до алгоритмів, OWASP визначає практичні підходи до їх реалізації та архітектури систем, а ISO забезпечує організаційний фундамент для впровадження цих вимог у діяльність підприємств. Усі ці стандарти вимагають комплексного, багаторівневого підходу до управління паролями, що включає застосування сучасних криптографічних алгоритмів, аналіз надійності паролів, забезпечення конфіденційності каналів зв'язку та впровадження багатофакторної автентифікації. Таким чином, успішне дотримання міжнародних норм є ключовою умовою створення надійної та стійкої системи управління паролями.

2.2 Основні підходи до побудови систем керування паролями

Системи керування паролями є критично важливим елементом інфраструктури інформаційної безпеки, оскільки забезпечують захищене зберігання, організацію та обробку конфіденційних автентифікаційних даних. Незважаючи на розвиток альтернативних методів автентифікації, таких як біометричні технології або апаратні ключі, паролі й надалі залишаються базовим механізмом підтвердження особи у більшості цифрових сервісів. Це зумовлює необхідність розроблення систем, здатних поєднати високий рівень криптографічного захисту з доступністю для користувачів [1].

Одним із ключових принципів, який лежить в основі сучасних менеджерів паролів, є застосування концепції Zero-Knowledge Encryption. Відповідно до цього підходу, шифрування даних відбувається локально на пристрої користувача, до їх передавання у хмару, а сервери сервісу ніколи не отримують доступу до відкритого вмісту сховища. Така модель використовується у Bitwarden, 1Password та NordPass, що підтверджується їх офіційною документацією й оглядами безпеки [20]. Концепція Zero-Knowledge широко

описана в академічних криптографічних джерелах, оскільки є фундаментальною для побудови систем конфіденційного зберігання даних [21].

Фундаментальну роль у побудові менеджерів паролів відіграють алгоритми шифрування та хешування, які забезпечують стійкість до атак перебором та відновлення. Для шифрування вмісту сховищ сучасні рішення застосовують алгоритми AES-256 та XChaCha20-Poly1305, ефективність яких підтверджено у відповідних міжнародних криптографічних стандартах та RFC 8439 [22; 23]. Для формування майстер-ключів використовуються алгоритми PBKDF2, bcrypt або Argon2id, при цьому останній рекомендований NIST як найбільш стійкий до атак за допомогою спеціалізованого апаратного забезпечення [28].

Надійність автентифікації у менеджерах паролів забезпечується завдяки застосуванню механізмів багатофакторної автентифікації (MFA). Використання додаткових факторів – таких як одноразові коди TOTP, апаратні ключі FIDO2/WebAuthn чи біометричні дані – відповідає рекомендаціям OWASP та вважається найкращою практикою в сучасних системах керування доступом [24; 25].

Сучасні системи керування паролями включають інструменти аналізу надійності паролів, що дозволяють користувачам уникати слабких або скомпрометованих комбінацій. Одним із найпоширеніших інструментів є алгоритм zxcvbn, розроблений Dropbox, який оцінює складність пароля на основі виявлення словникових збігів, типових патернів та ймовірних послідовностей символів [26]. Окрім цього, менеджери паролів інтегрують API сервісу HaveIBeenPwned, який використовує модель k-Anonymity для пошуку пароля у базах відомих витоків, не розкриваючи сам пароль або його повний хеш [27].

Важливим аспектом проєктування є також забезпечення механізмів відновлення доступу. Сучасні менеджери паролів застосовують аварійні ключі, секретні фрази, системи екстреного доступу та методи розподіленого зберігання секретів. Застосування таких механізмів відповідає принципам криптографічного захисту, викладеним у працях провідних дослідників сучасної криптографії [21].

Підсумовуючи, сучасні підходи до побудови систем керування паролями поєднують використання перевірених криптографічних алгоритмів, архітектурну гнучкість, суворе розмежування доступів та інтелектуальні інструменти аналізу. Усі ці компоненти спрямовані на забезпечення конфіденційності, цілісності та доступності облікових даних користувача в умовах зростання кіберзагроз і поширення складних атак.

2.3 Криптографічні методи захисту паролів

Сучасні системи управління паролями використовують широкий спектр криптографічних методів для забезпечення захисту автентифікаційних даних. Ефективність таких систем визначається не тільки складністю алгоритмів, а й їх правильною реалізацією та взаємодією між різними компонентами – шифруванням, хешуванням, управлінням ключами та перевіркою надійності паролів. Усі сучасні рекомендації міжнародних стандартів, включно з NIST SP 800-63B, наголошують, що пароль є одним із найбільш уразливих елементів у системах автентифікації, а тому застосування криптографічних механізмів є критично обов’язковим [28].

2.3.1 Алгоритми шифрування

Одним з основних напрямів криптографічного захисту є шифрування бази даних паролів. Це означає, що навіть у разі отримання зловмисником доступу до файлу сховища, зміст залишатиметься недоступним.

2.3.1.1 AES-256

Найчастіше використовується алгоритм AES-256 – стійкий симетричний алгоритм, властивості якого докладно описані у криптографічній літературі та працях Брюса Шнайєра [21]. Його привабливість полягає у тому, що він пройшов

багаторічну криптоаналітичну валідацію та є стандартом урядових структур США.

Алгоритм AES-256 є симетричним блочним шифром, що працює з блоком даних фіксованої довжини 128 біт та використовує секретний ключ довжиною 256 біт. Вхідний блок подається у вигляді матриці стану:

$$S = \left(s_{r,c}\right), \qquad r, c \in \{0, 1,2,3\}, \tag{2.1}$$

де кожен елемент $s_{r,c}$ є байтом, тобто елементом скінченного поля:

$$F_{2^8} = F_2[x]/\langle x^8 + x^4 + x^3 + x + 1\rangle \tag{2.2}$$

Шифрування AES-256 складається з 14 раундів і починається з операції додавання початкового раундового ключа:

$$S^{(0)} = P \oplus K^{(0)}, \tag{2.3}$$

де P – відкритий текст, $K^{(0)}$ – початковий ключ, а $\oplus$ означає побітове додавання за модулем 2. Для раундів $r = 1, \dots, 13$ перетворення має вигляд:

$$S^{(r)} = MixColumns(ShiftRows(SubBytes(S^{(r-1)}))) \oplus K^{(r)}, \tag{2.4}$$

а останній раунд описується формулою:

$$C = ShiftRows\left(SubBytes\left(S^{(13)}\right)\right) \oplus K^{(14)} \tag{2.5}$$

Операція SubBytes визначається як нелінійне перетворення кожного байту стану за формулою:

$$S(a) = A \times a^{-1} \oplus \mathrm{b}, \tag{2.6}$$

де a^{-1} – мультиплікативно обернений елемент у полі F_{2^8} (для $a = 0$ вважається $a^{-1} = 0$), A – фіксована матриця 8×8 над F_2, а b – сталий 8-бітний вектор. Перестановка ShiftRows реалізується циклічним зсувом рядків матриці стану на r позицій для r-го рядка.

Лінійне перетворення MixColumns виконується множенням кожного стовпця стану на фіксовану матрицю над F_{2^8}:

$$\begin{pmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{pmatrix} = \begin{pmatrix} 02 & 03\ 01 & 01 \\ 01 & 02\ 03 & 01 \\ 01 & 01\ 02 & 03 \\ 03 & 01\ 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{pmatrix} \tag{2.7}$$

Розгортання ключа AES-256 базується на формуванні послідовності слів W_i з початкових восьми 32-бітних слів ключа за рекурентним співвідношенням:

$$W_i = \begin{cases} W_{i-8} \oplus \mathrm{SubWord}\big(\mathrm{RotWord}(W_{i-1})\big) \oplus Rcon_{\frac{1}{8}}, & i = 0 (mod\ 8), \\ W_{i-8} \oplus \mathrm{SubWord}(W_{i-1}), & i = 4 (mod\ 8), \\ W_{i-8} \oplus (W_{i-1}), & \text{інакше.} \end{cases} \tag{2.8}$$

Раундові ключі формуються як $K^{(r)} = (W_{4r}, W_{4r+1}, W_{4r+2}, W_{4r+3})$. Дешифрування AES-256 здійснюється застосуванням обернених до наведених перетворень операцій у зворотному порядку раундів, що забезпечує коректне відновлення відкритого тексту за правильного секретного ключа.

2.3.1.2 XChaCha20-Poly1305

Поряд із ним дедалі більшого поширення набуває XChaCha20-Poly1305 – алгоритм потокового шифрування, що входить до стандарту RFC 8439 [23]. На відміну від AES, який є блочним шифром, XChaCha20 забезпечує стабільно

високий рівень продуктивності на різних платформах та має значно простішу реалізацію, що знижує ризик помилок програмування, які часто стають джерелом вразливостей.

Алгоритм XChaCha20-Poly1305 є симетричним автентифікованим алгоритмом шифрування, який поєднує потоковий шифр XChaCha20 та механізм автентифікації Poly1305. Він використовує секретний ключ довжиною 256 біт і nonce довжиною 192 біти, що значно зменшує ризик повторного використання параметрів шифрування. Шифрування здійснюється за схемою побітового додавання за модулем два між відкритим текстом і ключовим потоком.

Внутрішній стан ChaCha20 подається у вигляді матриці з 16 слів по 32 біти, над якою виконуються операції додавання за модулем 2^{32}, побітового XOR та циклічних зсувів. Базовою операцією алгоритму є quarter-round, що забезпечує нелінійність і дифузію бітів. Повний алгоритм складається з 20 раундів, після яких початковий стан додається до фінального для формування ключового потоку:

$$Keystream = State_{final} + State_{initial} \ (mod\ 2^{32}) \tag{2.9}$$

XChaCha20 розширює стандартний ChaCha20 шляхом використання функції HChaCha20, яка з секретного ключа та частини nonce формує підключ, що дозволяє безпечно застосовувати довгі nonce без зниження криптографічної стійкості.

Автентифікація повідомлення виконується за допомогою алгоритму Poly1305, який обчислює тег автентичності як поліноміальну функцію над скінченним полем:

$$tag = \left(\sum m_i\, r^i\right) mod\ (2^{130} - 5) \tag{2.10}$$

Отриманий тег дозволяє перевірити цілісність і автентичність зашифрованих даних під час розшифрування.

Таким чином, XChaCha20-Poly1305 поєднує прості арифметичні операції з доведеними криптографічними властивостями та забезпечує високий рівень захисту даних, що робить його придатним для використання в сучасних системах управління паролями.

2.3.2 Алгоритми хешування

Другим ключовим компонентом є криптографічне хешування майстер-пароля, який є центральним елементом системи. Якщо зловмисник зможе підібрати майстер-пароль, він отримає доступ до всіх інших паролів у сховищі. Тому варто виділити підходи до розподіленого зберігання секретів. Ця концепція дозволяє розділити майстер-ключ на кілька частин, кожна з яких зберігається різними сторонами. Такий механізм широко застосовується для корпоративних систем, де доступ до сховища може бути пов’язаний із кількома відповідальними особами.

Саме тому прості хеші на кшталт SHA-1 або SHA-256 не можуть забезпечити належний рівень захисту, оскільки ці алгоритми є надто швидкими й легко піддаються перебору на сучасних графічних процесорах. Для захисту майстер-пароля використовуються повільні, пам’яттєво-складні алгоритми змішування ключів: PBKDF2, bcrypt, scrypt та Argon2id.

2.3.2.1 PBKDF2

Алгоритм PBKDF2 базується на багаторазовому застосуванні криптографічної хеш-функції. Формально функція виведення ключа визначається як:

$$DK = PBKDF2(P, S, c, dkLen), \tag{2.11}$$

де P – пароль, S – сіль, c – кількість ітерацій, $dkLen$ – довжина вихідного ключа. Обчислення кожного блоку ключа виконується за схемою:

$$T_i = U_1 \oplus U_2 \oplus \ldots \oplus U_c, \tag{2.12}$$

де

$$U_1 = PRF(P, S \| i), \qquad U_j = PRF\left(P, U_{j-1}\right).$$

Збільшення параметра c лінійно підвищує обчислювальну складність алгоритму, однак PBKDF2 не є пам'яттєво-складним і тому менш стійкий до атак із використанням GPU.

2.3.2.2 bcrypt

Алгоритм bcrypt ґрунтується на блочному шифрі Blowfish і використовує адаптивний параметр вартості $cost$, який визначає кількість внутрішніх ітерацій:

$$Hash = bcrypt(P, S, cost) \tag{2.13}$$

Фактична кількість обчислень пропорційна 2^{cost}, що дозволяє легко масштабувати складність зі зростанням обчислювальних потужностей. Внутрішня структура bcrypt передбачає багаторазове розширення ключа Blowfish, що робить алгоритм стійким до простого перебору, проте його пам'яттєві вимоги залишаються відносно невеликими.

2.3.2.3 scrypt

Алгоритм scrypt був розроблений з метою протидії апаратним атакам і формально описується як:

$$DK = scrypt(P, S, N, r, p, dkLen), \tag{2.14}$$

де параметр N визначає розмір пам'яті, а r і p – параметри паралелізму.

Обчислювальна складність scrypt є пропорційною $O(N \cdot r)$, а пам'яттєва складність – $O(N \cdot r)$, що суттєво ускладнює реалізацію ефективних атак на GPU або ASIC через високі вимоги до оперативної пам'яті.

2.3.2.4 Argon2id

Сучасним стандартом для захисту паролів є алгоритм Argon2id, який поєднує властивості Argon2i та Argon2d. Формально функція має вигляд:

$$DK = Argon2id(P, S, t, m, p), \tag{2.15}$$

де t – кількість проходів, m – обсяг використовуваної пам'яті, p – рівень паралелізму. Алгоритм будує двовимірну матрицю блоків пам'яті та виконує послідовні й залежні від даних обчислення:

$$B_{ij} = G\left(B_{ij-1}, B_{ref}\right), \tag{2.16}$$

де G – криптографічна компресійна функція, а B_{ref} – псевдовипадково вибраний попередній блок. Така структура забезпечує одночасний захист від атак по сторонніх каналах і від масивно-паралельних атак із використанням спеціалізованого обладнання.

Argon2id у 2015 році переміг у конкурсі Password Hashing Competition, а його технічні характеристики, описані у фінальному звіті PHC [30], підтверджують високий рівень стійкості до атак із залученням паралельних обчислювальних структур, таких як GPU та FPGA.

Важливу роль відіграє також використання сільових значень (salt). Сіль – це випадковий набір символів, який додається до пароля перед хешуванням. Завдяки цьому навіть два однакові паролі матимуть різні хеші, що робить неможливим використання rainbow-table атак. Концепція солі давно є

стандартом безпеки і детально описана у документах OWASP Password Storage Cheat Sheet [32]. На відміну від солі, перець (pepper) є секретним параметром, який зберігається окремо від бази даних і застосовується додатково до пароля. Це створює додатковий рівень захисту, оскільки навіть якщо зловмисник отримає хеші, він не зможе виконати перебір без доступу до секретного перецьового значення.

2.3.3 Механізми перевірки надійності паролів

До найбільш поширених механізмів перевірки надійності паролів належать:

1. zxcvbn (Dropbox) – алгоритм, який аналізує словникові збіги, повтори, логічні патерни, типові клавіатурні послідовності та прогнозованість пароля [15].
2. Entropy-based scoring – метод, що оцінює ймовірність перебору пароля на основі математичної ентропії.
3. Пошук у базах витоків (NordPass, Bitwarden, 1Password) – механізми перевірки пароля на присутність у відомих витоках, що значно підвищує рівень захисту системи [19; 20].

Одним із найвідоміших інструментів є алгоритм zxcvbn, створений Dropbox. Його робота, технічні принципи та відкритий код доступні у репозиторії GitHub [35]. На відміну від традиційних методів, що лише аналізують довжину та склад символів, zxcvbn враховує людські поведінкові патерни – зокрема часті слова, дати, клавіатурні послідовності та поширені фрази. Таким чином, менеджери паролів можуть надати користувачу об’єктивну оцінку ризику.

Багато сучасних менеджерів паролів включають механізми перевірки надійності паролів, що використовують як математичні показники ентропії, так і аналіз шаблонів.

Ентропія – міра непередбачуваності комбінації символів. Згідно з рекомендаціями NIST SP 800-63B, збільшення довжини пароля підвищує його

стійкість у геометричній прогресії, навіть якщо структура пароля не змінилась [17]. Ентропія визначається формулою:

$$H = L \times \log_2 N, \tag{2.17}$$

де L – довжина пароля, N – кількість можливих символів у наборі.

Дослідження Security.org демонструють, що пароль із 12 символів є у мільйони разів стійкішим до перебору, ніж пароль із 6 символів, навіть якщо використовується той самий набір символів [18].

Таблиця 2.1 – Залежність ентропії від довжини пароля для набору 62 символів (A–Z, a–z, 0–9)

Довжина	**Ентропія**	**Сталість до перебору**
6 знаків	~35 біт	зламується за хвилини
8 знаків	~47 біт	години / дні
10 знаків	~60 біт	тижні / роки
14 знаків	~84 біти	століття

За даними з таблиці 2.1 можна створити графік залежності, що представлений на рисунку 2.1.

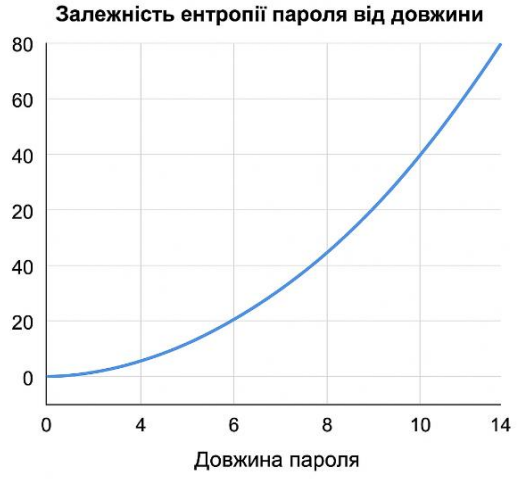

Рисунок 2.1 – Графік залежності ентропії від довжини пароля

Також одним із найважливіших сучасних механізмів є перевірка на витоки паролів. Найбільша база скомпрометованих паролів у світі – «Have I Been Pwned» – дозволяє здійснювати пошук за принципом k-Anonymity, коли користувач передає лише перші п’ять символів SHA-1-хешу пароля. Документація API пояснює, що цей метод дозволяє провадити перевірку без розкриття повного хешу [36]. Такий підхід став фактично стандартом для хмарних менеджерів паролів.

У таблиці 2.2 наведено порівняння криптографічних методів захисту паролів у сучасних системах.

Таблиця 2.2 – Порівняння криптографічних методів захисту паролів у сучасних системах

Категорія	**Метод**	**Переваги**	**Недоліки**	**Джерело**
Шифрування	AES-256	Висока стійкість, стандарт урядового рівня	Складна реалізація може містити помилки	[21]
Шифрування	XChaCha20-Poly1305	Простота, висока продуктивність, стійкість	Новіший стандарт, менше формальних доказів	[23]
Хешування	Argon2id	Найкращий захист від GPU-атак	Потребує більше ресурсів	[30]
Хешування	PBKDF2	Широко підтримуєтьс я у всіх системах	Вразливий до GPU при низьких параметрах	[28]
Перевірка витоків	HaveIBeenPwned API (k-Anonymity)	Перевірка без розкриття пароля	Лише SHA-1, залежність від зовнішнього сервісу	[36]

Продовження таблиці 2.2

Аналіз складності	zxcvbn	Виявляє шаблони та словникові збіги	Не оцінює криптографічну стійкість	[35]
Канали передачі	TLS 1.3	Найвища безпека серед TLS	Вимагає підтримки сучасних клієнтів	[34]
MFA	FIDO2/WebAuthn	Стійкість до фішингу та MITM	Потребує апаратних ключів	[33]

Системи керування паролями, що підтримують синхронізацію між пристроями, повинні також забезпечувати захист каналів передачі даних. Це досягається завдяки протоколу TLS 1.3, описаному у стандарті RFC 8446 [34]. TLS 1.3 усуває застарілі криптографічні методи та суттєво зменшує кількість можливих точок компрометації, усуваючи слабкі процедури рукостискання, властиві попереднім версіям протоколу. Крім цього, системи автентифікації використовують FIDO2 та WebAuthn, стандарти яких описані FIDO Alliance [33], що дозволяє застосовувати апаратні ключі безпеки для захисту доступу до менеджера паролів.

3 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ПАРОЛЯМИ

3.1 Вибір технологій для реалізації

Для реалізації системи управління паролями було обрано мову програмування Python, що зумовлено її високою швидкістю розроблення, широкою екосистемою криптографічних бібліотек та зручністю створення прототипів прикладних систем безпеки. Python активно використовується у сфері інформаційної безпеки, що робить його доцільним вибором для експериментальної реалізації криптографічних алгоритмів і подальшого аналізу їх ефективності.

Для створення графічного інтерфейсу користувача використано бібліотеку PyQt5, яка є Python-обгорткою фреймворку Qt. PyQt5 забезпечує кросплатформеність застосунку, підтримку сучасних елементів інтерфейсу та можливість відокремлення логіки програми від її візуального представлення. Застосування Qt Style Sheets дозволило реалізувати стилізацію інтерфейсу без використання сторонніх графічних бібліотек, що спростило структуру проєкту та підвищило зручність користування додатком.

Для забезпечення конфіденційності даних у системі використано симетричний алгоритм шифрування AES у режимі GCM, який поєднує шифрування з перевіркою цілісності даних. Вибір AES-GCM обґрунтований його широким застосуванням у сучасних стандартах захисту інформації, високою криптографічною стійкістю та апаратною оптимізацією на більшості сучасних процесорів. Реалізація алгоритму здійснюється за допомогою бібліотеки cryptography, яка є де-факто стандартом для криптографічних операцій у середовищі Python і забезпечує коректну та безпечну реалізацію алгоритмів.

Для захисту майстер-пароля застосовано алгоритм виведення ключа Argon2id, який поєднує властивості пам’яттєво-складних алгоритмів та стійкість до атак по сторонніх каналах. У разі недоступності Argon2id у середовищі

виконання використовується алгоритм PBKDF2 як резервний варіант. Такий підхід дозволяє забезпечити сумісність застосунку з різними середовищами та одночасно відповідати сучасним рекомендаціям щодо захисту паролів.

Для генерації випадкових значень, зокрема солі, nonce та паролів, використано модуль secrets, який забезпечує криптографічно стійке джерело випадковості. Це є критично важливим для коректної роботи як алгоритмів шифрування, так і алгоритмів виведення ключів. Зберігання даних організовано у вигляді зашифрованого файлу формату .vault, що містить серіалізовані структури даних у форматі JSON, зашифровані перед записом на носій.

Таким чином, обраний набір технологій забезпечує баланс між криптографічною стійкістю, практичною реалізовністю та зручністю використання. Використання Python і PyQt5 дозволило швидко реалізувати прототип з повноцінним графічним інтерфейсом, що показано на рисунку 3.1, а застосування сучасних криптографічних алгоритмів і бібліотек гарантує відповідність розробленої системи актуальним вимогам інформаційної безпеки.

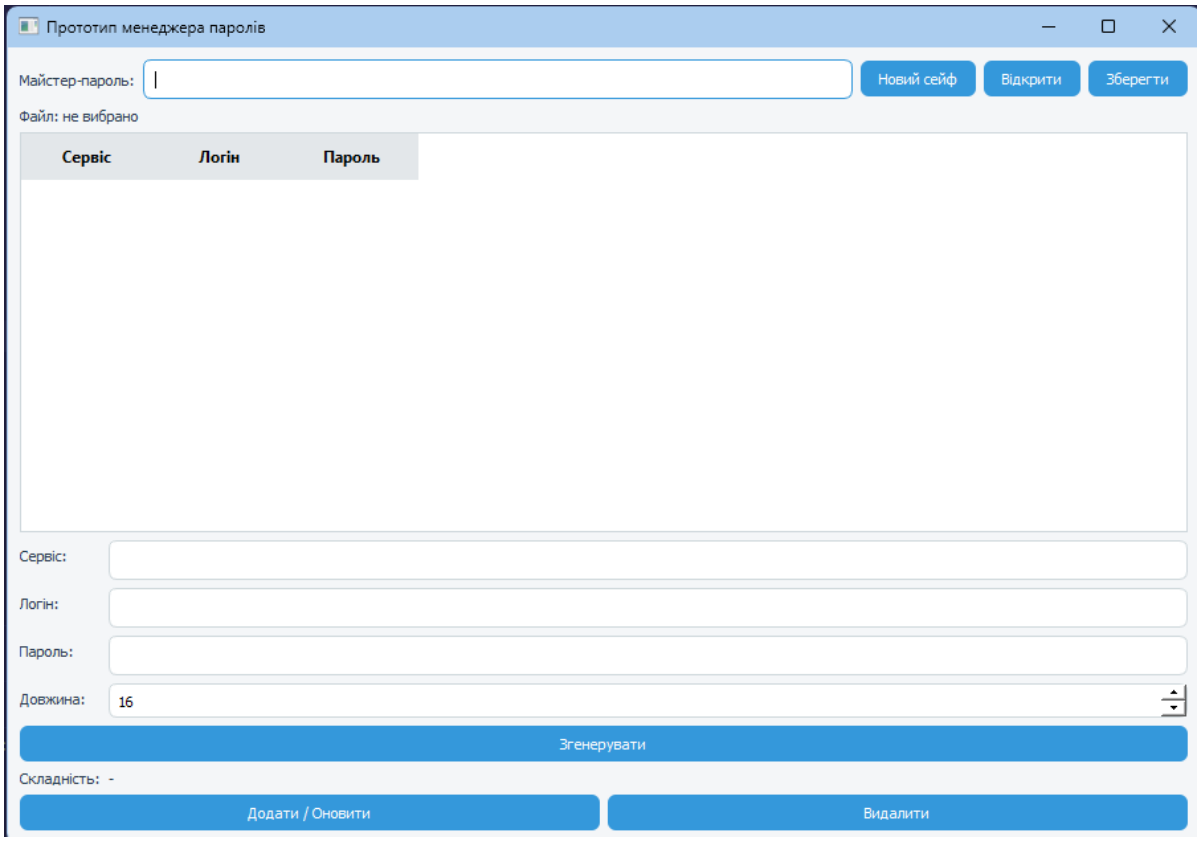

Рисунок 3.1 – Графічний інтерфейс менеджера паролів

3.2 Вибір криптографічних механізмів

Під час розроблення менеджера паролів ключовим завданням було забезпечення конфіденційності, цілісності та стійкості до атак на облікові дані користувача. З цією метою було обрано сучасні криптографічні механізми, які відповідають актуальним рекомендаціям у сфері захисту інформації та широко застосовуються у промислових рішеннях.

Для шифрування сховища паролів використано симетричний алгоритм AES у режимі GCM. Вибір AES-256 обумовлений його високою криптографічною стійкістю, стандартизацією та підтримкою апаратного прискорення на більшості сучасних процесорів. Режим GCM забезпечує не лише конфіденційність даних, а й їх цілісність та автентичність, оскільки під час шифрування формується тег автентифікації, який перевіряється на етапі розшифрування. Це дозволяє виявити будь-які спроби модифікації зашифрованого файлу сховища. У реалізації застосовується унікальне випадкове значення nonce для кожної операції шифрування, що виключає повторне використання параметрів і знижує ризик криптографічних атак.

Захист майстер-пароля реалізовано за допомогою алгоритмів виведення ключів із паролів, основним з яких є Argon2id. Даний алгоритм є пам’яттєво-складним і забезпечує високу стійкість до атак повного перебору з використанням GPU та спеціалізованих апаратних засобів. Argon2id поєднує переваги підходів Argon2i та Argon2d, що дозволяє одночасно протидіяти як масивно-паралельним атакам, так і атакам по сторонніх каналах. У разі відсутності підтримки Argon2id у середовищі виконання використовується резервний алгоритм PBKDF2, який базується на багаторазовому застосуванні криптографічної хеш-функції та дозволяє збільшити обчислювальну складність процесу виведення ключа шляхом налаштування кількості ітерацій.

У процесі збереження даних менеджер паролів формує зашифрований контейнер, у якому зберігається вся інформація про облікові записи користувача. Для демонстрації роботи криптографічних механізмів на рисунку 3.2 наведено

вміст файлу сховища, відкритого у текстовому редакторі без використання майстер-пароля. Як видно зі скріншота, дані представлені у вигляді випадкової на вигляд послідовності символів, що не містить жодної зрозумілої інформації про сервіси, логіни або паролі користувача. Це підтверджує, що перед записом на диск усі дані проходять процес шифрування з використанням алгоритму AES у режимі GCM, а ключ шифрування формується за допомогою алгоритму виведення ключа з майстер-пароля. Навіть за наявності доступу до файлу сховища без правильного ключа неможливо відновити вихідні дані, що наочно демонструє ефективність обраних криптографічних механізмів захисту.

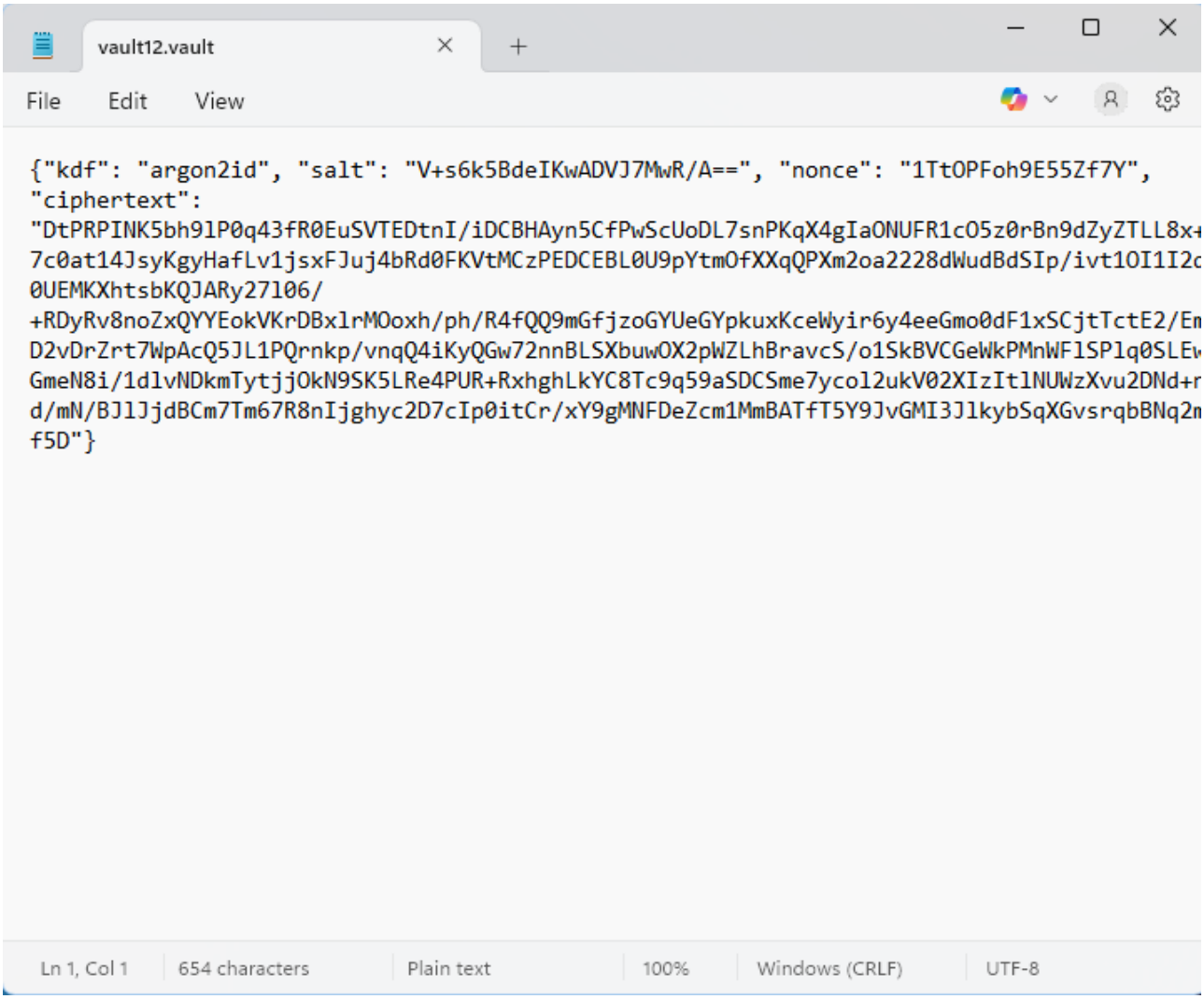

Рисунок 3.2 – Вміст зашифрованого файлу сховища паролів (.vault) без доступу до майстер-пароля

Хеш-функції в системі використовуються як складова алгоритмів виведення ключів, а не для прямого зберігання паролів. У складі PBKDF2 застосовується криптографічна хеш-функція SHA-256, яка забезпечує стійкість до колізій і передобразних атак у межах поставленої задачі. Такий підхід відповідає сучасним рекомендаціям, згідно з якими паролі не повинні зберігатися у вигляді простих хешів без додаткового ускладнення.

Генерація криптографічних ключів, сольових значень і параметрів nonce у системі здійснюється з використанням модуля secrets, який забезпечує криптографічно стійке джерело випадкових чисел. Це є критично важливим для коректної роботи як симетричних алгоритмів шифрування, так і алгоритмів виведення ключів, оскільки передбачуваність випадкових значень може призвести до компрометації всієї системи. Згенеровані сольові значення використовуються для унеможливлення атак із попередньо обчисленими таблицями, а випадкові nonce забезпечують безпеку режиму GCM.

3.3 Захист переданих і збережених даних

Захист збережених і потенційно переданих даних у розробленому менеджері паролів реалізовано з урахуванням принципів End-to-End Encryption та Zero-Knowledge, які застосовуються в сучасних промислових системах управління паролями. Реалізація цих підходів у програмному модулі дозволяє забезпечити конфіденційність даних незалежно від середовища їх зберігання або передачі.

У межах розробленого застосунку всі облікові дані користувача зберігаються виключно у зашифрованому вигляді у файлі формату .vault. Перед записом на диск структура даних, що містить список сервісів, логінів і паролів, серіалізується у формат JSON та шифрується на стороні клієнта з використанням алгоритму AES-256 у режимі GCM. Для кожної операції шифрування в коді генерується унікальне випадкове значення nonce, а результат шифрування містить також тег автентичності, який перевіряється під час розшифрування. Це

гарантує, що будь-які зміни файлу сховища будуть виявлені під час спроби доступу до даних.

Ключ шифрування в системі не зберігається у готовому вигляді, а динамічно формується під час відкриття або збереження сховища. У реалізації системи ключ генерується з майстер-пароля користувача за допомогою алгоритму Argon2id, який застосовує пам’яттєво-складний підхід і значно ускладнює атаки повного перебору. У випадку відсутності підтримки Argon2id використовується резервний механізм PBKDF2 із великою кількістю ітерацій. Таким чином, навіть при компрометації зашифрованого файлу сховища сторонній користувач не має можливості відновити ключ без знання майстер-пароля.

Принцип End-to-End Encryption у програмі реалізовано на рівні архітектури: усі криптографічні операції виконуються локально в межах клієнтського застосунку, а зовнішні компоненти не мають доступу до відкритих даних або ключів. Навіть у разі передачі файлу .vault через незахищені канали зв’язку, наприклад під час резервного копіювання або перенесення між пристроями, передається вже зашифрований контейнер, який не розкриває жодної інформації без відповідного майстер-пароля.

Для демонстрації реалізації принципу End-to-End Encryption на рисунку 3.3 наведено файл зашифрованого сховища паролів, збережений у файловій системі. Даний файл може бути перенесений між пристроями або використаний як резервна копія без ризику компрометації конфіденційних даних, оскільки всі облікові записи зберігаються у зашифрованому вигляді та не можуть бути прочитані без відповідного майстер-пароля. Файл сховища зберігається у вигляді єдиного контейнера, що не містить відкритих даних і не потребує захищеного каналу передачі.

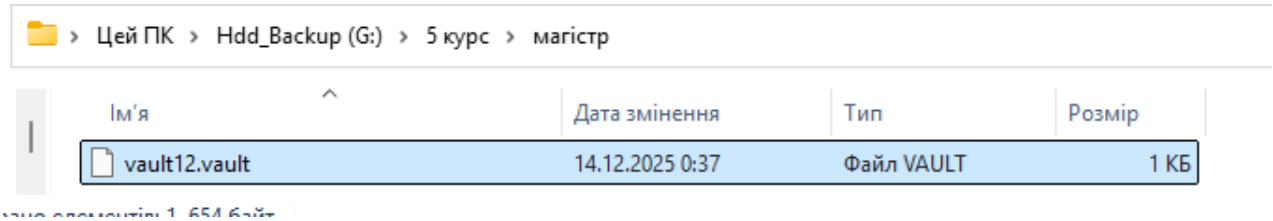

Рисунок 3.3 – Зашифрований файл сховища паролів (.vault), готовий до передачі або резервного зберігання

Концепція Zero-Knowledge у розробленому застосунку реалізується шляхом повної відсутності збереження майстер-пароля або його хешу у файловій системі чи конфігураційних файлах. У коді майстер-пароль використовується лише тимчасово в оперативній пам’яті для генерації ключа шифрування та після завершення операцій не зберігається. Єдиними додатковими параметрами, що записуються у файл сховища, є сіль і nonce, які є випадковими значеннями та не дозволяють відновити секретні ключі або паролі користувача. Таким чином, навіть розробник або адміністратор системи не має технічної можливості отримати доступ до збережених облікових даних, що відповідає моделі Zero-Knowledge.

Для підтвердження реалізації принципу Zero-Knowledge на рисунку 3.4 показано результат спроби відкриття зашифрованого сховища з використанням неправильного майстер-пароля.

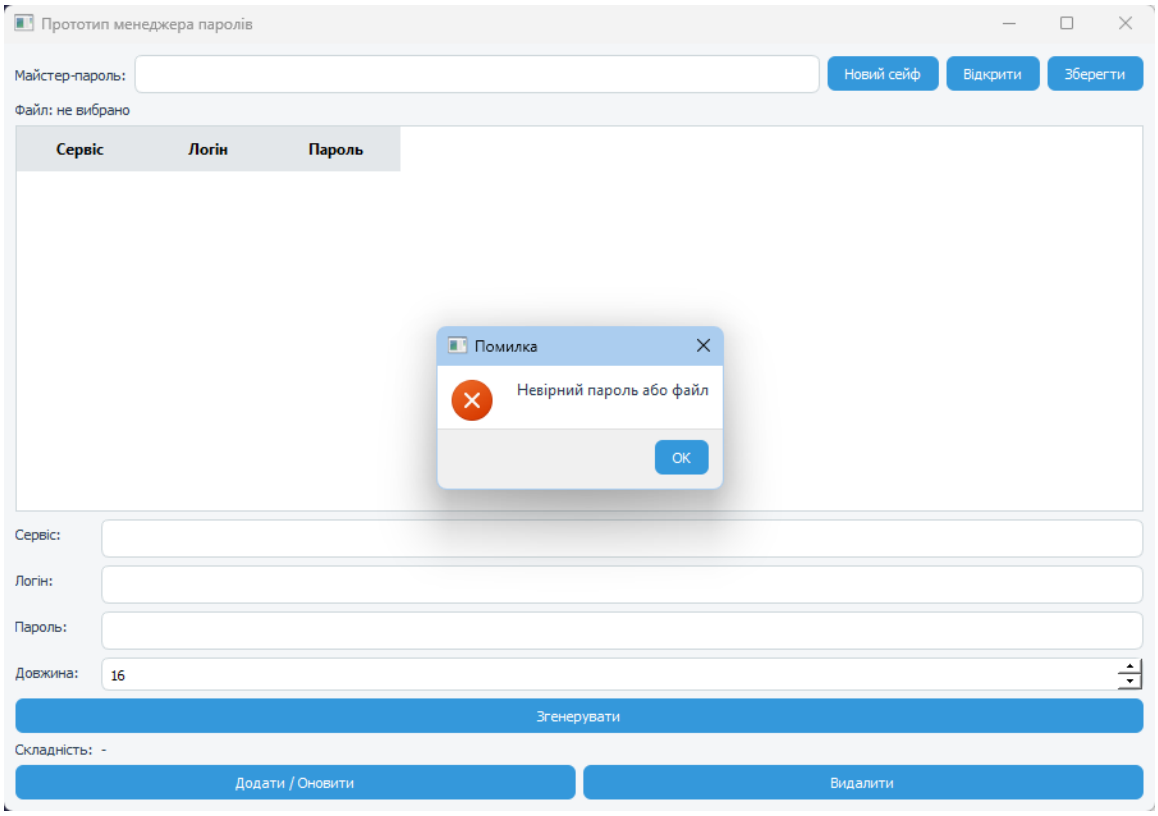

Рисунок 3.4 – Відмова в доступі до зашифрованого сховища при введенні неправильного майстер-пароля

Як видно зі скріншота, система не надає доступу до жодних облікових даних і завершує процес дешифрування з помилкою. Це свідчить про те, що майстер-пароль не зберігається у відкритому або хешованому вигляді та використовується виключно для формування ключа шифрування. У разі введення неправильного значення ключ не може бути відновлений, що повністю унеможливлює доступ до вмісту сховища.

Отже, реалізований у програмному застосунку підхід до захисту збережених і переданих даних поєднує криптографічні механізми та архітектурні рішення, які забезпечують практичну реалізацію принципів End-to-End Encryption і Zero-Knowledge. Це дозволяє гарантувати, що конфіденційні дані користувача залишаються захищеними навіть у разі компрометації середовища зберігання або каналу передачі, і підтверджує відповідність розробленої системи сучасним вимогам до безпечних менеджерів паролів.

3.4 Механізми автентифікації та багатофакторного захисту

Механізми автентифікації у розробленому менеджері паролів спрямовані на забезпечення контролю доступу до зашифрованого сховища та запобігання несанкціонованому використанню конфіденційних даних. Основним елементом автентифікації в системі є майстер-пароль користувача, який виконує роль єдиного секрету для доступу до всіх збережених облікових даних.

На рисунку 3.5 показано початковий етап автентифікації користувача в розробленому менеджері паролів.

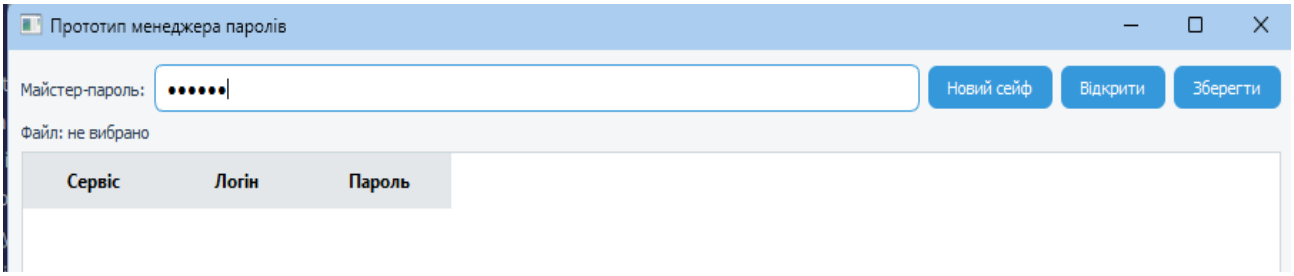

Рисунок 3.5 – Введення майстер-пароля як основного фактору автентифікації

Доступ до зашифрованого сховища можливий лише після введення майстер-пароля, який використовується для формування ключа шифрування.

У реалізованому застосунку автентифікація здійснюється шляхом введення майстер-пароля під час відкриття або створення сховища. Введений пароль безпосередньо не зберігається і не порівнюється з будь-яким еталонним значенням. Замість цього він використовується як вхідний параметр для алгоритму виведення ключа Argon2id, на основі якого формується ключ шифрування. У разі введення неправильного майстер-пароля процес дешифрування завершується помилкою перевірки тегу автентичності AES-GCM, що унеможливлює доступ до даних без коректного секрету. Такий підхід дозволяє реалізувати автентифікацію без необхідності зберігання хешів паролів або таблиць відповідності.

Результат успішної автентифікації користувача наведено на рисунку 3.6.

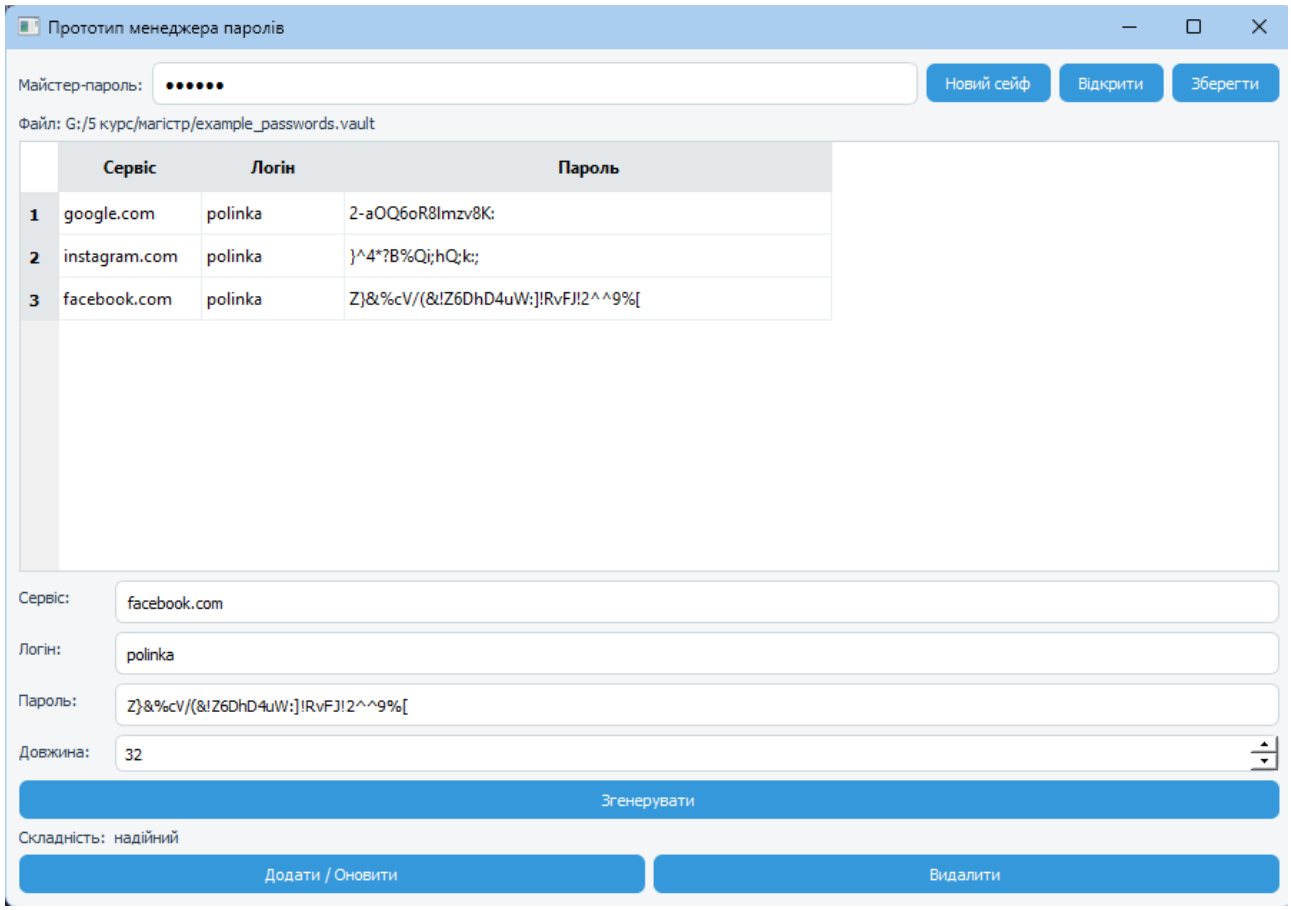

Рисунок 3.6 – Успішна автентифікація та доступ до зашифрованого сховища

Після введення коректного майстер-пароля формується правильний ключ шифрування, що дозволяє розшифрувати сховище та отримати доступ до збережених облікових даних.

Застосування пам’яттєво-складного алгоритму Argon2id додатково підвищує рівень захисту автентифікації, оскільки значно ускладнює атаки повного перебору навіть у разі отримання зловмисником зашифрованого файлу сховища. У коді прототипу передбачено резервний механізм на основі PBKDF2, що дозволяє забезпечити працездатність системи у середовищах, де Argon2id недоступний, зберігаючи при цьому прийнятний рівень безпеки.

Хоча розроблений прототип не реалізує повноцінну багатофакторну автентифікацію у вигляді апаратних токенів або одноразових кодів, його архітектура передбачає можливість розширення механізмів захисту. Зокрема, другий фактор автентифікації може бути реалізований у вигляді додаткового секрету, що вводиться користувачем, апаратного ключа або зовнішнього

підтвердження доступу. На концептуальному рівні таким фактором у прототипі виступає сам зашифрований файл сховища, доступ до якого є необхідною умовою для автентифікації разом із майстер-паролем.

Крім того, роль додаткового рівня захисту виконує механізм контролю цілісності даних, реалізований через режим AES-GCM. Будь-яка спроба модифікації зашифрованого файлу або підміни його вмісту призводить до неможливості успішної автентифікації та дешифрування, що фактично виконує функцію перевірки автентичності джерела даних. Таким чином, система захищена не лише від підбору майстер-пароля, а й від атак, спрямованих на підміну або пошкодження сховища.

Отже, у розробленому прототипі автентифікація базується на поєднанні знання користувача (майстер-пароль), криптографічно стійких алгоритмів виведення ключів та перевірки цілісності зашифрованих даних. Хоча багатофакторний захист реалізований на концептуальному рівні, обрана архітектура дозволяє легко інтегрувати додаткові фактори автентифікації в подальших версіях системи, що відповідає сучасним вимогам до безпечних менеджерів паролів.

3.5. Реалізація базового функціоналу системи

У розробленому прототипі менеджера паролів базовий функціонал реалізовано у вигляді окремих логічних модулів, інтегрованих у графічний інтерфейс на базі PyQt5. Керування роботою системи здійснюється через головне вікно застосунку, яке забезпечує доступ до основних операцій зі сховищем та обліковими записами.

Функція створення та відкриття сховища реалізується через механізми роботи з зашифрованим файлом формату .vault. Під час створення нового сховища формується початкова порожня структура даних, яка зберігається у пам’яті до моменту шифрування. При відкритті існуючого файлу користувач вводить майстер-пароль, на основі якого за допомогою алгоритму Argon2id

формується ключ шифрування. Отриманий ключ використовується для дешифрування вмісту файлу, після чого дані завантажуються у таблицю інтерфейсу. У разі введення неправильного пароля дешифрування завершується помилкою, і доступ до сховища не надається.

Робота з обліковими записами у системі реалізована через операції додавання, редагування та видалення записів безпосередньо з графічного інтерфейсу. Кожен запис містить назву сервісу, логін і пароль та відображається у таблиці головного вікна. Зміни, внесені користувачем, тимчасово зберігаються в оперативній пам’яті та застосовуються остаточно лише після виконання операції збереження, під час якої оновлена структура даних повторно шифрується та записується у файл сховища.

На рисунку 3.7 показано процес додавання нового облікового запису до зашифрованого сховища. Користувач вводить назву сервісу, логін і пароль, після чого дані тимчасово зберігаються у внутрішній структурі програми та набувають постійного характеру після збереження сховища. У прототипі також реалізовано генератор паролів, який дозволяє автоматично створювати надійні випадкові значення заданої довжини. Генерація здійснюється з використанням криптографічно стійкого джерела випадковості, що виключає використання передбачуваних або повторюваних символів. Згенерований пароль може бути одразу використаний для створення нового запису або заміни існуючого значення в таблиці.

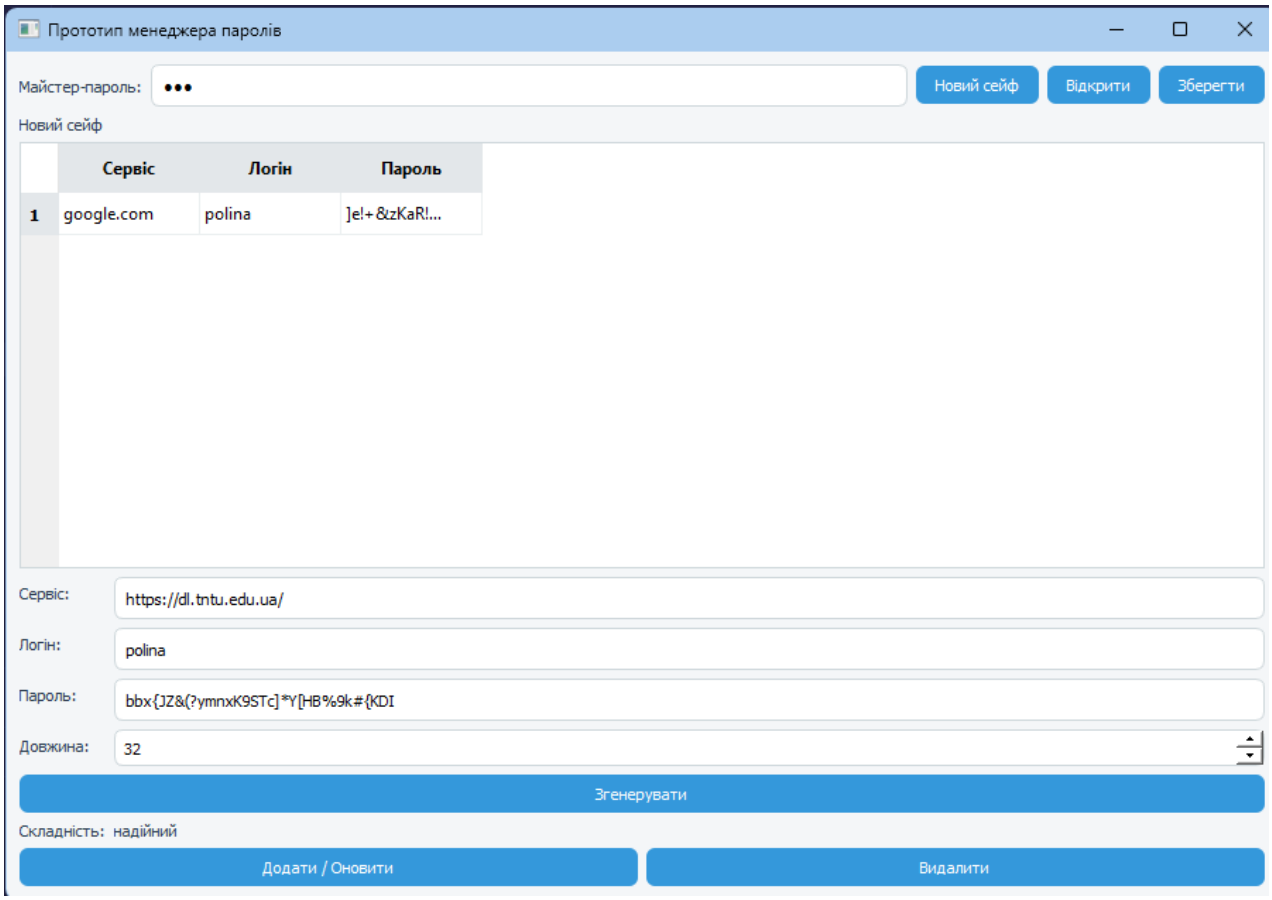

Рисунок 3.7 – Додавання нового облікового запису до сховища паролів

Для підвищення рівня безпеки користувацьких даних у коді реалізовано механізм оцінки складності паролів. Під час введення або генерації пароля система аналізує його довжину та склад символів і відображає користувачу узагальнену інформацію про надійність обраного значення. Окремо передбачено квазі-реалізацію перевірки паролів на наявність у відомих списках витоків, що дозволяє попередити використання слабких або скомпрометованих паролів без прямого звернення до зовнішніх сервісів. Для підвищення рівня безпеки облікових даних у системі реалізовано генератор паролів. На рисунку 3.8 наведено приклад автоматично згенерованого пароля, створеного з використанням криптографічно стійкого джерела випадковості, що унеможливлює передбачуваність значень.

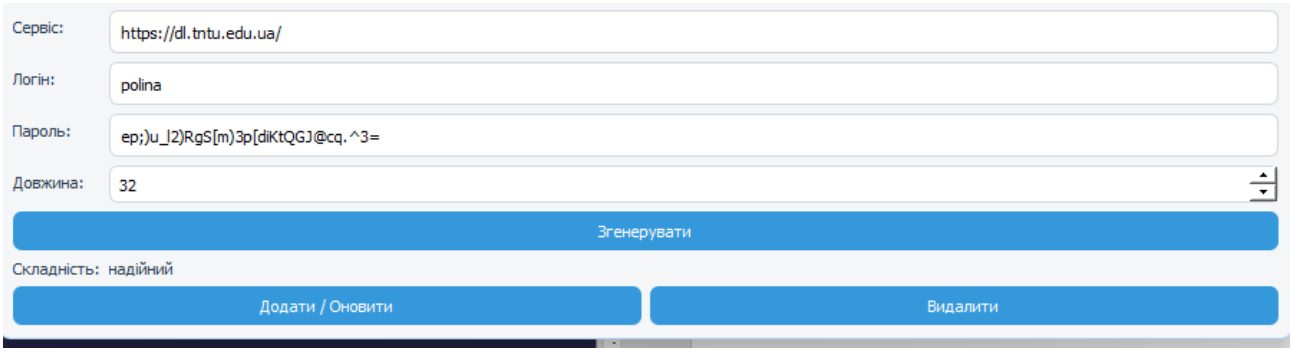

Рисунок 3.8 – Генерація випадкового надійного пароля

Інтерфейс користувача побудований таким чином, щоб забезпечити мінімальну кількість дій для виконання типових операцій. Візуальне оформлення реалізоване з використанням стилів Qt Style Sheets, що дозволило виділити основні елементи керування та покращити сприйняття інформації. Усі критичні операції, пов’язані з доступом до сховища, супроводжуються перевірками коректності введених даних та повідомленнями про помилки, що зменшує ймовірність некоректного використання системи.

Таким чином, реалізація базового функціоналу прототипу менеджера паролів поєднує практичні механізми роботи з обліковими даними та сучасні криптографічні підходи. Конкретні програмні рішення, застосовані у коді, демонструють можливість створення безпечного та зручного інструмента для управління паролями в межах настільного застосунку.

3.6 Тестування безпеки та функціональності

Тестування розробленого прототипу менеджера паролів проводилося з метою перевірки коректності реалізації основного функціоналу, а також оцінки рівня захищеності системи від типових загроз інформаційної безпеки. Процес тестування включав функціональні перевірки, а також сценарії, спрямовані на виявлення можливих вразливостей, пов’язаних із доступом до зашифрованих даних.

Функціональне тестування передбачало перевірку основних сценаріїв використання системи. Було перевірено створення нового зашифрованого

сховища, відкриття існуючого файлу .vault, додавання, редагування та видалення облікових записів, а також коректність збереження даних після повторного відкриття сховища. Усі зазначені операції виконувалися без помилок, а дані зберігалися у цілісному вигляді після завершення роботи програми та її повторного запуску. Окремо перевірено роботу генератора паролів, який стабільно створював випадкові значення заданої довжини з використанням криптографічно стійкого джерела випадковості.

Тестування механізмів автентифікації полягало у спробах доступу до сховища з використанням як правильного, так і неправильного майстер-пароля. У разі введення коректного пароля доступ до даних надавався успішно, тоді як введення неправильного значення призводило до помилки дешифрування та повної відмови в доступі. Це підтверджує коректну роботу алгоритмів виведення ключа та режиму шифрування AES-GCM, а також відсутність збереження будь-яких допоміжних даних, які могли б бути використані для обходу автентифікації.

Окрему увагу було приділено тестуванню стійкості системи до несанкціонованого доступу до файлу сховища. Для цього здійснювалися спроби відкриття .vault-файлу без використання програми, а також спроби його модифікації вручну. У всіх випадках файл залишався непридатним для читання, а будь-яке порушення його цілісності призводило до неможливості успішного розшифрування. Це підтверджує ефективність реалізації захисту цілісності даних, що забезпечується режимом AES-GCM.

Також було перевірено роботу механізмів оцінки складності паролів і квазі-перевірки на наявність у списках скомпрометованих значень. Система коректно реагувала на використання слабких або поширених паролів, попереджаючи користувача про потенційні ризики, що сприяє підвищенню загального рівня безпеки облікових даних.

Для систематизації результатів тестування функціональності та безпеки розробленого прототипу менеджера паролів було сформовано перелік основних сценаріїв перевірки, що охоплюють як типові операції користувача, так і потенційні загрози несанкціонованого доступу. Тестування проводилося шляхом

моделювання реальних умов експлуатації системи, зокрема створення та відкриття зашифрованого сховища, управління обліковими записами, а також спроб доступу до даних без коректної автентифікації або з порушенням цілісності файлу сховища. Результати виконаних тестів, очікувані та фактичні наслідки кожного сценарію узагальнено в таблиці 3.1.

Таблиця 3.1 – Результати тестування безпеки та функціональності прототипу менеджера паролів

№	**Сценарій тестування**	**Опис дії**	**Очікуваний результат**	**Фактичний результат**
1	Створення нового сховища	Створення нового файлу .vault із заданим майстер-паролем	Формування порожнього зашифрованого сховища	Сховище створено успішно
2	Відкриття сховища з правильним паролем	Введення коректного майстер-пароля	Надання доступу до збережених даних	Доступ надано
3	Відкриття сховища з неправильним паролем	Введення некоректного майстер-пароля	Відмова в доступі до даних	Доступ заблоковано
4	Перегляд файлу сховища	Відкриття .vault у текстовому редакторі	Неможливість прочитати вміст	Дані зашифровані
5	Модифікація файлу сховища	Зміна байтів у .vault вручну	Неможливість розшифрування	Дешифрування неможливе
6	Додавання нового запису	Додавання сервісу, логіну та пароля	Запис додається до сховища	Запис додано
7	Видалення запису	Видалення облікового запису зі сховища	Запис видаляється	Запис видалено

Продовження таблиці 3.1

8	Генерація пароля	Автоматичне створення пароля	Генерується випадковий складний пароль	Пароль згенеровано
9	Оцінка складності пароля	Введення слабкого пароля	Попередження про низьку надійність	Попередження відображено
10	Перевірка на витоки	Введення поширеного пароля	Повідомлення про ризик	Ризик виявлено

Наведені в таблиці результати тестування підтверджують коректність реалізації базового функціоналу та механізмів захисту інформації в розробленому прототипі. Усі перевірені сценарії завершилися очікуваним результатом, що свідчить про відсутність критичних помилок у роботі системи та ефективність застосованих криптографічних механізмів.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОХОРОНА ПРАЦІ

4.1 Охорона праці у процесі розроблення та впровадження програмної системи управління паролями

При розробленні та впровадженні програмної системи управління паролями враховано вимоги охорони праці, технічної та протипожежної безпеки відповідно до чинного законодавства України та сучасних міжнародних стандартів. Це зумовлено тим, що процеси створення, тестування та експлуатації програмного забезпечення безпосередньо пов’язані з тривалою роботою персоналу за комп’ютерною технікою, що створює додаткові фізичні та психоемоційні навантаження. У сучасних умовах інформаційна безпека програмних систем не може розглядатися ізольовано від безпеки праці фахівців, які забезпечують їх функціонування, адміністрування та підтримку.

Відповідно до Закону України «Про охорону праці» роботодавець зобов’язаний забезпечити безпечні та здорові умови праці, організувати робочі місця з урахуванням вимог безпеки та здійснювати контроль за дотриманням нормативних вимог у сфері охорони праці [37]. Ці положення безпосередньо стосуються діяльності, пов’язаної з розробленням і використанням інформаційних систем, зокрема програмних засобів для управління паролями, оскільки їх експлуатація передбачає постійну роботу з комп’ютерною технікою.

Кодекс законів про працю України визначає обов’язок створення безпечних і нешкідливих умов праці на всіх підприємствах та в установах, незалежно від форми власності [38]. У контексті впровадження програмної системи управління паролями це означає необхідність організації робочих місць розробників і користувачів таким чином, щоб мінімізувати негативний вплив тривалої роботи за комп’ютером, знизити ризик професійних захворювань і помилок, спричинених перевтомою.

Закон України «Про забезпечення санітарного та епідемічного благополуччя населення» встановлює обов’язок роботодавців забезпечувати

належний стан виробничого середовища та контролювати вплив фізичних факторів на здоров'я працівників [39]. Для персоналу, який працює з програмними системами інформаційної безпеки, це передбачає дотримання вимог щодо мікроклімату приміщень, освітлення, рівня шуму та електромагнітного випромінювання від комп'ютерної техніки.

Важливою складовою охорони праці є пожежна безпека. Відповідно до чинних Правил пожежної безпеки в Україні, приміщення, у яких розміщується комп'ютерна техніка та серверне обладнання, повинні бути оснащені первинними засобами пожежогасіння, а електричні мережі та обладнання – відповідати встановленим вимогам безпеки [40]. Це є критично важливим для забезпечення безпечної експлуатації програмних систем, у яких зберігаються та обробляються конфіденційні дані.

У міжнародному контексті питання охорони праці регламентуються стандартом ISO 45001:2018, який визначає вимоги до систем управління охороною праці та безпекою персоналу [41]. Згідно з цим стандартом, організації повинні ідентифікувати ризики, пов'язані з використанням інформаційних технологій, та впроваджувати заходи для їх зменшення. Це положення є актуальним для процесу розроблення програмної системи управління паролями, оскільки помилки персоналу або перевтома можуть призвести до зниження рівня інформаційної безпеки.

Крім того, міжнародні стандарти ISO/IEC 27001:2022 та ISO/IEC 27002:2022 підкреслюють важливість людського фактора в системах управління інформаційною безпекою та необхідність створення безпечних умов праці для персоналу, який має доступ до інформаційних ресурсів [42], [43]. Це підтверджує, що охорона праці є невід'ємною складовою комплексного підходу до забезпечення інформаційної безпеки програмних систем.

Таким чином, у процесі розроблення та впровадження програмної системи управління паролями враховано актуальні вимоги охорони праці, технічної та пожежної безпеки. Дотримання чинного законодавства України та сучасних міжнародних стандартів дозволяє забезпечити безпечні умови праці персоналу,

знизити вплив людського фактора та створити надійну основу для ефективного функціонування системи управління інформаційною безпекою.

4.2 Інженерний захист персоналу об'єкту та населення. Правила застосування

Інженерний захист персоналу об’єкту та населення є одним із базових елементів системи цивільного захисту та спрямований на зменшення негативного впливу небезпечних факторів надзвичайних ситуацій техногенного, природного та воєнного характеру. Відповідно до Кодексу цивільного захисту України, інженерний захист включає комплекс технічних і будівельних заходів, спрямованих на забезпечення безпеки життя і здоров’я людей у разі виникнення надзвичайних ситуацій [44].

Основним завданням інженерного захисту є створення умов, за яких персонал об’єкта та населення мають можливість уникнути або мінімізувати вплив уражальних факторів, таких як ударна хвиля, уламки будівельних конструкцій, пожежі, радіаційне випромінювання, токсичні хімічні речовини та інші небезпечні впливи. Для цього застосовуються захисні споруди цивільного захисту, інженерні укріплення будівель, а також спеціальні технічні рішення, спрямовані на підвищення стійкості об’єктів інфраструктури [45].

До основних елементів інженерного захисту належать захисні споруди цивільного захисту, зокрема сховища та протирадіаційні укриття. Вимоги до таких споруд визначені державними будівельними нормами, які регламентують їх конструктивні характеристики, місткість, рівень захисту, наявність систем вентиляції, водопостачання, аварійного освітлення та санітарно-гігієнічних умов [46]. Захисні споруди повинні забезпечувати можливість перебування людей протягом встановленого часу без загрози для їх життя та здоров’я.

Важливим напрямом інженерного захисту є підвищення стійкості будівель і споруд до дії небезпечних факторів. Це досягається шляхом використання міцних будівельних матеріалів, додаткового укріплення несучих конструкцій,

захисту віконних і дверних прорізів, а також раціонального розміщення обладнання та інженерних комунікацій. Планування території об’єкта повинно здійснюватися з урахуванням можливих зон ураження та забезпечення безпечних шляхів евакуації персоналу і населення [45], [47].

Окрему роль в інженерному захисті відіграє захист інженерних мереж та систем життєзабезпечення. Надійність електропостачання, водопостачання, вентиляції та систем зв’язку є критично важливою для функціонування об’єкта в умовах надзвичайної ситуації. Відповідно до рекомендацій міжнародних організацій, резервування та дублювання інженерних систем значно підвищує рівень безпеки та знижує ризик повного виходу об’єкта з ладу [48].

Правила застосування інженерного захисту передбачають не лише створення відповідних технічних умов, а й постійну підтримку захисних споруд у готовності до використання. Це включає регулярні перевірки технічного стану укриттів, контроль справності інженерних систем, оновлення планів евакуації та інструкцій для персоналу. Відповідні заходи повинні здійснюватися відповідно до нормативних документів та методичних рекомендацій органів цивільного захисту [47].

Крім того, ефективність інженерного захисту значною мірою залежить від рівня підготовки персоналу та населення. Навчання правилам користування захисними спорудами, знання розташування укриттів і дотримання встановлених алгоритмів дій у надзвичайних ситуаціях є необхідною умовою зменшення людських втрат. Міжнародний досвід у сфері управління ризиками надзвичайних ситуацій свідчить, що поєднання інженерних рішень та організаційних заходів є найбільш ефективним підходом до захисту населення [47], [48].

Таким чином, інженерний захист персоналу об’єкта та населення є комплексною системою заходів, що охоплює проектування та утримання захисних споруд, підвищення стійкості будівель, захист інженерних комунікацій і забезпечення готовності людей до дій у надзвичайних ситуаціях. Реалізація зазначених заходів відповідно до чинних нормативних вимог і

міжнародних рекомендацій дозволяє суттєво знизити рівень ризику та мінімізувати наслідки надзвичайних ситуацій для життя і здоров’я людей [41]–[48].

ВИСНОВКИ

У межах кваліфікаційної роботи було проаналізовано сучасні підходи до управління паролями та основні загрози, пов’язані з їх зберіганням і використанням. На основі проведеного аналізу розроблено прототип менеджера паролів, який поєднує сучасні криптографічні алгоритми та практичні механізми захисту інформації.

У роботі реалізовано систему захищеного зберігання облікових даних із використанням алгоритмів AES-256 та Argon2id, що забезпечує конфіденційність, цілісність і стійкість до атак повного перебору. Архітектура прототипу відповідає принципам End-to-End Encryption та Zero-Knowledge, що унеможливлює доступ до паролів без знання майстер-пароля навіть у разі компрометації файлу сховища.

Проведене тестування підтвердило коректність реалізації базового функціоналу та ефективність обраних криптографічних механізмів. Отримані результати свідчать про практичну придатність розробленого прототипу та доцільність використання запропонованих рішень для створення безпечних систем управління паролями.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fernando, W.P.K., Dissanayake, D.A.N.P., Dushmantha, S.G.V.D., Liyanage, D.L.C.P., & Karunatilake, C. (2023). Challenges and opportunities in password management: A review of current solutions. Sri Lanka Journal of Social Sciences and Humanities, 3(2), 9-20.

2. HIPAA Journal. Improper use of password managers is increasing Improper use of password managers is increasing [Електронний ресурс]. – Режим доступу: https://www.hipaajournal.com/improper-use-of-password-managers-is-increasing/

3. Dropbox. Zxcvbn: password strength estimator Zxcvbn: password strength estimator [Електронний ресурс]. – Режим доступу: https://github.com/dropbox/zxcvbn

4. Have I Been Pwned. K-Anonymity model description Have I Been Pwned: Pwned Passwords API and k-Anonymity model [Електронний ресурс]. – Режим доступу: https://haveibeenpwned.com/API/v3#PwnedPasswords

5. Password entropy and strength models (NIST SP 800-63B) NIST Special Publication 800-63B: Digital Identity Guidelines – Authentication and Lifecycle Management [Електронний ресурс]. – Режим доступу: https://pages.nist.gov/800-63-3/sp800-63b.html

6. Security.org. Password Entropy Explained Password entropy explained [Електронний ресурс]. – Режим доступу: https://www.security.org/how-secure-is-my-password/password-entropy/

7. NordPass Password Check System NordPass Security. Password health & breach detection system overview [Електронний ресурс]. – Режим доступу: https://nordpass.com/features/password-health/

8. Bitwarden Audit & Breach Detection Bitwarden Security Audit & Data Breach Detection Features [Електронний ресурс]. – Режим доступу: https://bitwarden.com/help/security-reports/

9. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. – Wiley, 2015.

10. Bernstein D. J. The Poly1305-AES Message-Authentication Code // Workshop Record of SAC 2004.

11. Langley A., Hamburg M., Turner S. RFC 8439: ChaCha20 and Poly1305 for IETF Protocols [Електронний ресурс]. – Режим доступу: https://datatracker.ietf.org/doc/rfc8439/

12. OWASP Authentication Cheat Sheet [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org/

13. WebAuthn/FIDO2 Overview [Електронний ресурс]. – Режим доступу: https://webauthn.guide/

14. Dropbox. Zxcvbn Password Strength Estimator [Електронний ресурс]. – Режим доступу: https://github.com/dropbox/zxcvbn

15. Have I Been Pwned – API Documentation [Електронний ресурс]. – Режим доступу: https://haveibeenpwned.com/API/v3

16. NIST SP 800-63B. Digital Identity Guidelines [Електронний ресурс]. – Режим доступу: https://pages.nist.gov/800-63-3/

17. Haber S., Stornetta W. Secure timestamping // Journal of Cryptology. – 1991.

18. Khovratovich D., Biryukov A. Argon2: The Memory-Hard Function for Password Hashing. – PHC Final Report, 2015.

19. Bernstein D. J. ChaCha – a variant of Salsa20 [Електронний ресурс]. – https://cr.yp.to/chacha.html

20. OWASP Password Storage Cheat Sheet [Електронний ресурс]. – https://cheatsheetseries.owasp.org

21. FIDO Alliance. FIDO2 Specifications [Електронний ресурс]. – https://fidoalliance.org/specifications/

22. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3

23. Dropbox. Zxcvbn Password Strength Estimator – https://github.com/dropbox/zxcvbn

24. Hunt T. Have I Been Pwned – Password API – https://haveibeenpwned.com/API/v3

25. Про охорону праці : Закон України від 14 жовт. 1992 р. № 2694-XII : станом на 2024 р. – URL: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 16.12.2025).

26. Кодекс законів про працю України : Закон України від 10 груд. 1971 р. № 322-VIII : станом на 2024 р. – URL: https://zakon.rada.gov.ua/laws/show/322-08 (дата звернення: 16.12.2025).

27. Про забезпечення санітарного та епідемічного благополуччя населення : Закон України від 24 лют. 1994 р. № 4004-XII : станом на 2024 р. – URL: https://zakon.rada.gov.ua/laws/show/4004-12 (дата звернення: 16.12.2025).

28. Правила пожежної безпеки в Україні : наказ М-ва внутр. справ України від 30 груд. 2014 р. № 1417 : станом на 2024 р. – URL: https://zakon.rada.gov.ua/laws/show/z0252-15 (дата звернення: 16.12.2025).

29. ISO 45001:2018. Occupational health and safety management systems – Requirements with guidance for use. – Geneva : ISO, 2018.

30. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. – Geneva : ISO/IEC, 2022.

31. ISO/IEC 27002:2022. Information security controls. – Geneva : ISO/IEC, 2022.

32. Karpinski, M., Sokolov, A., Tokkuliyeva, A., Radush, V., Kazakova, N., Shaikhanova, A., ... & Korchenko, A. (2025). Development of High-Quality Cryptographic Constructions Based on Many-Valued Logic Affine Transformations. Electronics, 14(10), 2094.

33. M. Derkach, et al., CrypticWave: A Zero-Persistence Ephemeral Messaging System with Client-Side Encryption, in: Cyber Security and Data Protection, vol. 4042, 2025, 316–323.

34. Деркач М. В., Мишко О. Є. Використання алгоритму шифрування AES-256-CBC для зберігання даних автентифікації автономного помічника. Наукові вісті Далівського університету. 2023. №24.

35. Yesin, V., Karpinski, M., Yesina, M., Vilihura, V., Kozak, R., & Shevchuk, R. (2023). Technique for Searching Data in a Cryptographically Protected SQL Database. Applied Sciences, 13(20), 11525.

36. Skarga-Bandurova, I., Biloborodova, T., Kjelstrup-Johnson, K. R., Scheper, T. V. O., & Derkach, M. (2026). Securing Tomorrow's Cities: Smart Infrastructure for Emergency Response, Crisis Management, and Defence. In Sustainable, Innovative, and Intelligent Industries and Societies (pp. 69-111). Cham: Springer Nature Switzerland.

37. Zagorodna, N., Skorenkyy, Y., Kunanets, N., Baran, I., & Stadnyk, M. (2022). Augmented Reality Enhanced Learning Tools Development for Cybersecurity Major. In ITTAP (pp. 25-32).

38. Muzh, V., & Lechachenko, T. (2024). Computer technologies as an object and source of forensic knowledge: challenges and prospects of development. Вісник Тернопільського національного технічного університету, 115(3), 17-22.

39. Кодекс цивільного захисту України : Закон України від 02.10.2012 № 5403-VI (зі змінами).

40. ДБН В.2.2-5:2023. Захисні споруди цивільного захисту. Основні положення.

41. Державна служба України з надзвичайних ситуацій. Методичні рекомендації з організації цивільного захисту населення.

42. ISO 22301:2019. Security and resilience – Business continuity management systems – Requirements.

43. UN Office for Disaster Risk Reduction (UNDRR). Guidelines for Disaster Risk Reduction and Engineering Protection.

ДОДАТОК А

Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

17-18 грудня 2025 року

ТЕРНОПІЛЬ
2025

Продовження додатку А

УДК 004.056
П. Слободян; М. Богач; М. Деркач, канд. техн. наук, доц.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ ПОПУЛЯРНИХ СИСТЕМ УПРАВЛІННЯ ПАРОЛЯМИ

UDC 004.056
P. Slobodian; M. Bohach; M. Derkach, Ph.D., Assoc. Prof.

ANALYSIS OF POPULAR PASSWORD MANAGEMENT SYSTEMS

Сучасний ринок систем управління паролями пропонує великий спектр рішень, призначених для різних категорій користувачів – від приватних осіб до великих корпоративних організацій. Основна мета таких систем полягає у забезпеченні надійного та зручного зберігання облікових даних, захисту від несанкціонованого доступу та спрощенні процесу користування великою кількістю облікових записів. Незважаючи на загальну спрямованість на безпеку, менеджери паролів значно відрізняються між собою за архітектурою, функціональністю, способами синхронізації та рівнем захисту.

Одним із найпоширеніших менеджерів паролів є Bitwarden, який пропонує широкий спектр функцій, таких як автоматичне заповнення форм для входу на сайти, генерація складних паролів, спільний доступ до облікових даних у межах сім'ї або команди, а також інтеграцію з різними браузерами та мобільними платформами. В свою чергу, KeePass – це локальний менеджер паролів, який зберігає всі дані на пристрої користувача. 1Password орієнтований на комерційних користувачів та корпоративні потреби, пропонуючи зручний інтерфейс та комплексний набір функцій безпеки. Також сучасним хмарним рішенням є NordPass, що пропонує можливість зберігати не лише паролі, а й різні конфіденційні дані, наприклад, нотатки або інформацію про платіжні картки.

Таблиця 1. Аналіз популярних систем управління паролями

Характеристика	Bitwarden	KeePass	1Password	NordPass
Зберігання	Хмара/локально	Локально	Хмара	Хмара
Тип коду	Відкритий	Відкритий	Закритий	Закритий
Шифрування	AES-256	AES-256, Twofish	AES-256 + Secret Key	XChaCha20
Синхронізація	Так	Ні (вручну)	Так	Так
MFA	Так (TOTP, U2F)	Через плагіни	Так	Так
Перевірка витоків	Так	Через плагіни	Так	Так

Всебічний аналіз цих систем показує суттєві відмінності у підходах до організації даних, зручності використання та безпеки. Локальні менеджери, такі як KeePass, забезпечують максимальний контроль над інформацією, однак вимагають додаткових зусиль для резервного копіювання та синхронізації. Хмарні системи, такі як Bitwarden і NordPass, спрощують користування завдяки синхронізації на кількох пристроях і хмарному зберіганню, проте безпека даних значною мірою залежить від надійності серверної інфраструктури та застосованих криптографічних методів. 1Password поєднує зручність і безпеку, пропонуючи розширені корпоративні функції та високий рівень шифрування. Це різноманіття дозволяє користувачам обирати систему, яка найкраще відповідає їхнім технічним можливостям, потребам та вимогам до конфіденційності.

ДОДАТОК Б

Код додатку менеджера паролів

```
import json
import os
import sys
import base64
import secrets
import string
from datetime import datetime

from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
    QFormLayout, QLineEdit, QPushButton, QLabel, QFileDialog,
    QTableWidget, QTableWidgetItem, QMessageBox, QSpinBox, QCheckBox
)
from PyQt5.QtCore import Qt

from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

try:
    from argon2.low_level import hash_secret_raw, Type as Argon2Type
    HAS_ARGON2 = True
except ImportError:
    HAS_ARGON2 = False

# КРИПТОГРАФІЧНІ ФУНКЦІЇ

def derive_key_pbkdf2(password: str, salt: bytes, length: int = 32) ->
bytes:
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=length,
        salt=salt,
```

Продовження додатку Б

```
        iterations=200_000,
    )
    return kdf.derive(password.encode())


def derive_key_argon2id(password: str, salt: bytes, length: int = 32) -
> bytes:
    if not HAS_ARGON2:
        return derive_key_pbkdf2(password, salt, length)

    return hash_secret_raw(
        secret=password.encode(),
        salt=salt,
        time_cost=3,
        memory_cost=64_000,
        parallelism=2,
        hash_len=length,
        type=Argon2Type.ID
    )


def encrypt_vault(data: dict, password: str) -> bytes:
    plaintext = json.dumps(data, ensure_ascii=False).encode()
    salt = secrets.token_bytes(16)
    nonce = secrets.token_bytes(12)

    if HAS_ARGON2:
        key = derive_key_argon2id(password, salt)
        kdf_name = "argon2id"
    else:
        key = derive_key_pbkdf2(password, salt)
        kdf_name = "pbkdf2"

    aesgcm = AESGCM(key)
```

Продовження додатку Б

```
    ciphertext = aesgcm.encrypt(nonce, plaintext, None)

    wrapper = {
        "kdf": kdf_name,
        "salt": base64.b64encode(salt).decode(),
        "nonce": base64.b64encode(nonce).decode(),
        "ciphertext": base64.b64encode(ciphertext).decode()
    }
    return json.dumps(wrapper).encode()


def decrypt_vault(blob: bytes, password: str) -> dict:
    wrapper = json.loads(blob.decode())
    salt = base64.b64decode(wrapper["salt"])
    nonce = base64.b64decode(wrapper["nonce"])
    ciphertext = base64.b64decode(wrapper["ciphertext"])

    if wrapper["kdf"] == "argon2id" and HAS_ARGON2:
        key = derive_key_argon2id(password, salt)
    else:
        key = derive_key_pbkdf2(password, salt)

    aesgcm = AESGCM(key)
    plaintext = aesgcm.decrypt(nonce, ciphertext, None)
    return json.loads(plaintext.decode())

# ГЕНЕРАТОР ПАРОЛІВ ТА ПСЕВДО-ПЕРЕВІРКА ВИТОКІВ

BREACHED_PASSWORDS = {
    "123456", "password", "qwerty", "111111",
    "123456789", "letmein", "admin"
}
```

Продовження додатку Б

```
def generate_password(length=16, lower=True, upper=True, digits=True,
symbols=True):
    chars = ""
    if lower: chars += string.ascii_lowercase
    if upper: chars += string.ascii_uppercase
    if digits: chars += string.digits
    if symbols: chars += "!@#$%^&*()-_=+[]{};:,.?/"

    return "".join(secrets.choice(chars) for _ in range(length))


def password_strength(password: str) -> str:
    if password in BREACHED_PASSWORDS:
        return "КОМПРОМЕТОВАНИЙ"

    score = 0
    score += len(password) >= 8
    score += len(password) >= 12
    score += any(c.islower() for c in password)
    score += any(c.isupper() for c in password)
    score += any(c.isdigit() for c in password)
    score += any(c in "!@#$%^&*()-_=+[]{};:,.?/" for c in password)

    if score <= 2:
        return "дуже слабкий"
    elif score <= 4:
        return "середній"
    return "надійний"

# ГРАФІЧНИЙ ІНТЕРФЕЙС

class PasswordManagerWindow(QMainWindow):
    def __init__(self):
        super().__init__()
```

Продовження додатку Б

```
        self.setWindowTitle("Прототип менеджера паролів")
        self.resize(900, 600)

        self.vault = {"created": datetime.now().isoformat(), "entries":
[]}
        self.filepath = None

        self.init_ui()

    def init_ui(self):
        central = QWidget()
        layout = QVBoxLayout()
        central.setLayout(layout)
        self.setCentralWidget(central)

        top = QHBoxLayout()
        self.master = QLineEdit()
        self.master.setEchoMode(QLineEdit.Password)

        self.btn_new = QPushButton("Новий сейф")
        self.btn_open = QPushButton("Відкрити")
        self.btn_save = QPushButton("Зберегти")

        top.addWidget(QLabel("Майстер-пароль:"))
        top.addWidget(self.master)
        top.addWidget(self.btn_new)
        top.addWidget(self.btn_open)
        top.addWidget(self.btn_save)
        layout.addLayout(top)

        self.info = QLabel("Файл: не вибрано")
        layout.addWidget(self.info)

        self.table = QTableWidget(0, 3)
```

Продовження додатку Б

```
        self.table.setHorizontalHeaderLabels(["Сервіс", "Логін",
"Пароль"])
        layout.addWidget(self.table)

        form = QFormLayout()
        self.service = QLineEdit()
        self.login = QLineEdit()
        self.password = QLineEdit()

        self.length = QSpinBox()
        self.length.setRange(8, 64)
        self.length.setValue(16)

        self.btn_gen = QPushButton("Згенерувати")
        self.strength = QLabel("-")

        form.addRow("Сервіс:", self.service)
        form.addRow("Логін:", self.login)
        form.addRow("Пароль:", self.password)
        form.addRow("Довжина:", self.length)
        form.addRow(self.btn_gen)
        form.addRow("Складність:", self.strength)

        layout.addLayout(form)

        bottom = QHBoxLayout()
        self.btn_add = QPushButton("Додати / Оновити")
        self.btn_del = QPushButton("Видалити")

        bottom.addWidget(self.btn_add)
        bottom.addWidget(self.btn_del)
        layout.addLayout(bottom)

        self.btn_new.clicked.connect(self.new_vault)
```

Продовження додатку Б

```
        self.btn_open.clicked.connect(self.open_vault)
        self.btn_save.clicked.connect(self.save_vault)
        self.btn_gen.clicked.connect(self.gen_password)
        self.btn_add.clicked.connect(self.add_entry)
        self.btn_del.clicked.connect(self.delete_entry)
        self.password.textChanged.connect(self.update_strength)

    def new_vault(self):
        self.vault = {"created": datetime.now().isoformat(), "entries":
[]}
        self.filepath = None
        self.table.setRowCount(0)
        self.info.setText("Новий сейф")

    def open_vault(self):
        path, _ = QFileDialog.getOpenFileName(self, "Відкрити", "",
"*.vault")
        if not path:
            return
        try:
            with open(path, "rb") as f:
                self.vault = decrypt_vault(f.read(),
self.master.text())
            self.filepath = path
            self.refresh()
            self.info.setText(f"Файл: {path}")
        except Exception:
            QMessageBox.critical(self, "Помилка", "Невірний пароль або
файл")

    def save_vault(self):
        if not self.filepath:
            self.filepath, _ = QFileDialog.getSaveFileName(self,
"Зберегти", "vault.vault", "*.vault")
```

Продовження додатку Б

```
    if not self.filepath:
        return
    with open(self.filepath, "wb") as f:
        f.write(encrypt_vault(self.vault, self.master.text()))
    self.info.setText(f"Файл: {self.filepath}")

def gen_password(self):
    pwd = generate_password(self.length.value())
    self.password.setText(pwd)

def update_strength(self):
    self.strength.setText(password_strength(self.password.text()))

def add_entry(self):
    entry = {
        "service": self.service.text(),
        "login": self.login.text(),
        "password": self.password.text()
    }
    self.vault["entries"].append(entry)
    self.refresh()

def delete_entry(self):
    row = self.table.currentRow()
    if row >= 0:
        del self.vault["entries"][row]
        self.refresh()

def refresh(self):
    self.table.setRowCount(len(self.vault["entries"]))
    for i, e in enumerate(self.vault["entries"]):
        self.table.setItem(i, 0, QTableWidgetItem(e["service"]))
        self.table.setItem(i, 1, QTableWidgetItem(e["login"]))
        self.table.setItem(i, 2, QTableWidgetItem(e["password"]))
```

Продовження додатку Б

```
# ЗАПУСК

app = QApplication(sys.argv)
app.setStyleSheet("""
QMainWindow {
    background-color: #f4f6f8;
    font-family: Segoe UI;
    font-size: 13px;
}

QLabel {
    color: #2c3e50;
}

QLineEdit {
    background-color: #ffffff;
    border: 1px solid #cfd8dc;
    border-radius: 6px;
    padding: 6px;
}

QLineEdit:focus {
    border: 1px solid #3498db;
}

QPushButton {
    background-color: #3498db;
    color: white;
    border: none;
    border-radius: 6px;
    padding: 7px 14px;
}
```

Продовження додатку Б

```
QPushButton:hover {
    background-color: #2980b9;
}

QPushButton:pressed {
    background-color: #1f6391;
}

QTableWidget {
    background-color: #ffffff;
    border: 1px solid #cfd8dc;
    gridline-color: #ecf0f1;
}

QHeaderView::section {
    background-color: #e3e7ea;
    padding: 6px;
    border: none;
    font-weight: bold;
}

QSpinBox {
    background-color: #ffffff;
    border: 1px solid #cfd8dc;
    border-radius: 6px;
    padding: 4px;
}
""")
window = PasswordManagerWindow()
window.show()
app.exec_()
```