ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«WEB-ДОДАТОК З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ WEBAUTHN ДЛЯ БЕЗПАРОЛЬНОЇ АУТЕНТИФІКАЦІЇ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

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

_____ Єгор ДЕМЧЕНКО
(підпис) *(Ім'я ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти
гр. ІСДМ-61 _____ Єгор ДЕМЧЕНКО
(ім'я, ПРІЗВИЩЕ)

Керівник: Валентина ДАНИЛЬЧЕНКО
PhD (доктор філософії) *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти магістр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ Каміла
СТОРЧАК
«____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Демченко Єгор Юрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Web-додаток з використанням технології WebAuthn для безпарольної аутентифікації.
керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, PhD
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету
інформаційно-комунікаційних технологій від « 30 » жовтня 2025 р. №467

2. Строк подання кваліфікаційної роботи « 26 » грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи:
1. завдання на кваліфікаційну роботу студента
2. наукові статті
3. технічна література

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз та огляд предметної області.
2. Постановка задачі та методологія дослідження.
3. Розробка запропонованого методу безпарольної аутентифікації та засобів реалізації.
4. Реалізація, дослідження та результати.

5. Перелік ілюстративного матеріалу: презентація на слайдах

    1. Архітектура безпарольної аутентифікації у WebAuthn
    2. WebAuthn API: аутентифікація
    3. Етапи аутентифікації у WebAuthn
    4. Реалізація аутентифікації на практиці

6. Дата видачі завдання « 30 » жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	30.10.2025	Викон.
2	Написання 1 розділу кваліфікаційної роботи	8.11.2025	Викон.
3	Написання 2 розділу кваліфікаційної роботи	14.11.2025	Викон.
4	Написання 3 розділу кваліфікаційної роботи	23.11.2025	Викон.
5	Написання 4 розділу кваліфікаційної роботи	27.11.2025	Викон.
6	Висновки, вступ, реферат	30.11.2025	Викон.
7	Перевірка кваліфікаційної роботи на оригінальність тексту	01.12.2025	Викон.
8	Розробка презентації	10.12.2025	Викон.

Здобувач вищої освіти _____ Єгор ДЕМЧЕНКО
(підпис) *(Ім'я ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій
ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Демченко Є.Ю. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)
на тему: «Web-додаток з використанням технології WebAuthn для безпарольної аутентифікації»

Кваліфікаційна робота і рецензія додаються.

Директора ННІТ _____ Катерина НЕСТЕРЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

За час роботи над кваліфікаційною роботою здобувач Демченко Єгор Юрійович продемонстрував високий рівень теоретичної та дослідницької підготовки, успішно розглянувши сучасний метод безпарольної аутентифікації WebAuthn та його архітектуру для концептуальної розробки веб-додатку з використанням безпарольної технології дослідження WebAuthn з подальшим аналізом недоліків та переваг даної технології.

Він якісно провів аналіз архітектури безпарольної технології WebAuthn, обґрунтував засоби реалізації та отримав практично значущі результати, виявивши при цьому ініціативність та хист до наукової діяльності. З огляду на глибину опрацювання матеріалу та практичну цінність роботи, Демченко Єгор Юрійович заслуговує на оцінку «відмінно» та присвоєння кваліфікації магістр з інформаційних систем та технологій.

Керівник кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 2025 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Демченко Є.Ю. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедри Інформаційних систем та технологій _____ Каміла СТОРЧАК
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«____» _____ 2025 року

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня магістра

здобувача вищої освіти Демченко Єгор Юрійович
(прізвище, ім'я, по батькові)
на тему «Web-додаток з використанням технології WebAuthn для безпарольної аутентифікації»

Актуальність.
Актуальність дослідження зумовлена зростанням кількості кіберзагроз, спрямованих на компрометацію облікових даних користувачів. Оскільки паролі залишаються вразливими до кібератак, що створює ризики для користувачів і власників веб-сервісів. WebAuthn розглядається як перспективний стандарт безпарольної автентифікації, який здатний підвищити рівень інформаційної безпеки та покращити користувацький досвід. Концептуальний аналіз цієї технології є важливим для розуміння її потенціалу та можливостей застосування у сучасних веб-додатках.

Позитивні сторони.
1. Робота ретельно розглядає сучасні підходи безпарольної аутентифікації для забезпечення якісного досвіду користувача та підвищення безпеки даних, що демонструє високий дослідницький рівень.
2. Досліджується застосування безпарольної технології WebAuthn для легкої та безпечної аутентифікації, що підвищує теоретичну значущість дослідження.
3. Сформульована та реалізовані способи реалізації аутентифікації у концептуальній моделі веб-додатку, що є цінним практичним результатом, підтвердженим порівняльним аналізом отриманих теоретичних знань з подальшим розглядом переваг та недоліків безпарольної технології аутентифікації WebAuthn .

Недоліки.
1. Недостатньо детально розглянуто питання вразливостей технології безпарольної аутентифікації WebAuthn до кібератак.
2. Запропонованих рішень для розробки веб-додатку не достатньо для повноцінного планування та впровадження повністю функціонуючого веб-додатку, було б доцільно більш ретельно розглянути тему архітектури веб-дадатків.

Відзначені зауваження не впливають на загальну позитивну оцінку магістерської кваліфікаційної роботи

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "відмінно", а здобувач Демченко Єгор Юрійович заслуговує присвоєння кваліфікації: магістр з інформаційних систем та технологій.*

Рецензент _____ _____

науковий ступінь, вчене звання (підпис) *(Ім'я ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 71 стор., 4 рис., 1 таблиця, 20 джерела.

Метою є дослідження технології WebAuthn, як сучасного підходу до безпечної безпарольної аутентификации у веб-додатках. Робота спрямована на аналіз принципів функціонування WebAuthn, а також оцінку доцільності інтеграції цієї технології у сучасні веб-системи з точки зору безпеки, зручності використання та перспектив подальшого розвитку.

Об’єкт дослідження – це системи аутентифікації та авторизації у веб-додатках, що використовуються для підтвердження особи користувача та забезпечення контролю доступу до інформаційних ресурсів.

Предмет дослідження – це технологія WebAuthn та її архітектура, механізми роботи. Також концептуальні підходи до інтеграції WebAuthn у веб-додатки та взаємодії з існуючими протоколами аутентифікації.

Актуальність дослідження зумовлена зростанням кількості кіберзагроз, спрямованих на компрометацію облікових даних користувачів. Оскільки паролі залишаються вразливими до кібератак, що створює ризики для користувачів і власників веб-сервісів. WebAuthn розглядається як перспективний стандарт безпарольної автентифікації, який здатний підвищити рівень інформаційної безпеки та покращити користувацький досвід. Концептуальний аналіз цієї технології є важливим для розуміння її потенціалу та можливостей застосування у сучасних веб-додатках.

КЛЮЧОВІ СЛОВА: WEBAUTHN, FIDO2, БЕЗПАРОЛЬНА АУТЕНТИФІКАЦІЯ, ІНФОРМАЦІЙНА БЕЗПЕКА, ВЕБ-ДОДАТКИ, КРИПТОГРАФІЧНІ КЛЮЧІ, КІБЕРАТАКА.

ABSTRACT

Text part of the qualification work for obtaining a master's degree: 71 pages, 4 figures, 1 table, 20 sources.

The aim is to study WebAuthn technology as a modern approach to secure passwordless authentication in web applications. The work is aimed at analyzing the principles of WebAuthn functioning, as well as assessing the feasibility of integrating this technology into modern web systems in terms of security, usability, and prospects for further development.

The object of the study is authentication and authorization systems in web applications used to confirm the identity of users and ensure control of access to information resources.

The subject of the study is WebAuthn technology and its architecture and mechanisms of operation. It also covers conceptual approaches to integrating WebAuthn into web applications and interacting with existing authentication protocols.

The paper considers the relevance of applying computer vision to automate traffic monitoring. It analyzes the advantages and disadvantages of modern deep learning architectures used for object detection. The YOLOv8 architecture is investigated and its modification is proposed by introducing an attention module to improve the extraction of important features. A method for an integrated system capable of operating in real time is developed. An experimental study of the effectiveness and accuracy of the proposed model compared to baseline solutions is conducted.

KEYWORDS: WEBAUTHN, FIDO2, PASSWORDLESS AUTHENTICATION, INFORMATION SECURITY, WEB APPLICATIONS, CRYPTOGRAPHIC KEYS.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні веб-додатки стають все більш розповсюдженими для бізнесу, фінансових сервісів, державних структур та особистого користування, що робить питання безпечної аутентифікації важливим для кібербезпеки кібербезпеки. Звичайні паролі виявляються ненадійними через слабкі комбінації, повторне використання та фішингові атаки. Водночас зростає потреба у зручних та безпечних безпарольних методах автентифікації. Технологія WebAuthn, яка входить до стандарту FIDO2, пропонує сучасний підхід до забезпечення безпечного безпарольного доступу. Ця технологія відкриває нові перспективи для захисту веб-додатків.

Мета і завдання дослідження. Метою даного дослідження є концептуальна розробка веб-додатку з інтеграцією технології WebAuthn. Ааналіз принципів роботи архітектурних підходів та потенційних переваг перед типовими методами аутентифікації.

Об'єкт дослідження. Об’єктом дослідження є сучасні веб-додатки та механізми аутентифікації користувачів, що забезпечують доступ до інформаційних ресурсів.

Завдання:

1. Дослідити поняття аутентифікації, її роль у забезпеченні інформаційної безпеки та класичні методи авторизації.
2. Проаналізувати недоліки традиційних систем паролів і обґрунтувати необхідність переходу до безпарольних технологій.
3. Оглянути стандарти FIDO2, CTAP та WebAuthn і визначити їх місце в екосистемі сучасних веб-технологій.
4. Розглянути архітектуру WebAuthn, механізми створення та перевірки криптографічних пар ключів, формати даних і типи автентифікаторів.

5. Проаналізувати безпекові аспекти WebAuthn, сумісність із сучасними браузерами та операційними системами.
6. 0Розробити план розробки веб-додатку з інтеграцією WebAuthn, о0писати основні компоненти системи та основні алгоритми автентифікації.
7. Оцінити ефективність та безпечність системи, порівняти її з типовими методами і іншими безпарольними технологіями.
8. Дослідити перспективи розвитку стандарту WebAuthn та можливості його впровадження у комерційних і державних веб-системах.

Предмет дослідження. Предметом дослідження виступає технологія WebAuthn як інструмент безпечної аутентифікації її взаємодія з клієнтською та серверною частинами веб-додатку, а також інтеграція з сучасними протоколами управління ідентичністю.

Методи дослідження. Для досягнення поставлених цілей застосовано комплекс теоретичних методів. Тобто аналіз наукової літератури та стандартів FIDO2, систематизація існуючих методів аутентифікації, моделювання архітектури веб-додатків, концептуальна розробка схем взаємодії веб-додатка.

Практичне значення одержаних результатів. Результати дослідження мають практичне значення для розробників веб-додатків, служб безпеки та організацій, які прагнуть впровадити безпечну та зручну аутентифікацію.

Апробація:

І Всеукраїнська науково-технічна конференція «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу», на тему «Інтеграція технології автентифікації WebAuthn у web-додатках», що відбулася 18 листопада 2025 року.

1. ТЕОРЕТИЧНІ ОСНОВИ АУТЕНТИФІКАЦІЇ У ВЕБ-ДОДАТКАХ ТА ЇЇ РОЗВИТОК

1.1 Поняття аутентифікації та авторизації у веб-додатках

Аутентифікація важливий елемент впровадження безпеки для системи. Вона дозволяє переконатись, що доступ до системи отримає саме той користувач чи сервіс, який дійсно має всі права для доступу. У веб-додатках автентифікація дозволяє запобігати несанкціонаваним втручанням, контролюючі доступ до сервісів. Такий різновид захисту даних використовується у багатьох веб-сервісах.

Аутентифікація — це процес перевірки особи або системи з метою підтвердження їхньої легітимності перед доступом до інформаційних ресурсів.

Веб-додаток — це програмний застосунок, який впроваджений на веб-сервері та доступний користувачам через інтерфейс браузера. На відміну від програм, які не працюють через браузер, веб-додаток не вимагає бути встановленим на пристрій. Оскільки до нього можна отримати доступ через браузер, звичайно якщо він підтримується ним. Такий додаток потребує з'єднання з мережею інтернет, щоб повноцінно функціонувати. Для створення інтерфейсу зазвичай використовуються ті ж інструменти, що і для створення звичайного сайту. Тобто такі моги програмування як HTML, CSS та JavaScript. Логіка здебільшого реалізується через серверну частину. Сервер виконує операції з обробки даних, а клієнтська частина відповідає за взаємодію з користувачем. Це допомагає надавати послуги не залежно від операційної системи чи пристрою.

Авторизація — це процес визначення того, який рівень доступу користувач має і які дії він може виконувати після успішної аутентифікації в систему.

Після успішної аутентифікації, яка визначає ким є користувач для системи, потрібно визначити що йому дозволено в ній робити. Авторизація виглядає закономірним процесом після аутентифікації, під час якого визначається які дозволи має користувач. Користувач може виконувати і на якому рівні йому дозволено взаємодіяти з системою. У веб-додатках цей механізм зазвичай реалізується як контрольована система доступу до ресурсу і даних, де попередньо визначаються роль та можливості користувача. Після підтвердження особи сервер може сформувати потрібний маркер доступу або інший ідентифікатор сеансу, який зберігає дані про дозволи для користувача. Якщо користувач хоче виконати певну дію в системі, то вона спочатку перевірить чи входить така дія до списку дозволених для нього. Наприклад при спробі переглядання конфіденційних даних чи ініціації адміністративної операції система повинна упевнетись, що подібні дії дозволені для цього користувача.

Авторизація часто реалізується за допомогою наступних найбільш поширених моделей для ролей:

- RBAC (Role-Based Access Control) – це модель контролю доступу, у якій користувач отримує не прямі дозволи, а певні ролі. Вони містять конкретний набір правил, який і визначає можливості користувача. Тому з'являється можливість керувати доступом систематизовано. У великих компаніях модель зазвичай використовується та хмарних середовищах.
- ABAC (Attribute-Based Access Control) — модель контролю доступу в якій рішення про надання прав приймаються на основі певних атрибутів користувача. Атрибутами можуть бути певні власті користувача чи середовища. Тут не використовуються визначені ролі, а замість цього система аналізує певний набір характеристик користувача і порівнює їх з політикою доступу.

Такий підхід до авторизації дає можливість створити гнучку та динамічну систему, яка гарно підходить для складних систем управління, що дозволяє більш точно розподіляти дозволи.

Принцип логіки RBAC концептуально доволі простий. Спочатку організації варто визначити ролі, які їм потрібні. Після чого призначити їх певним робітникам або службам. Створюється модель із визначеною структурою, в якій користувач має певну ролі, а ця роль отримує конкретні дозволи. Новий співробітник автоматично може отримати доступ до тих дій, які відповідають його посаді. Якщо співробітник покидає свою посаду, буде достатньо просто позбавити його ролі яку він до цього мав. Модель RBAC має змогу непогано масштабуватись в різних інформаційних системах. За потреби внесення змін в політику безпек адміністратору не потрібно взаємодіяти окремо з кожним користувачем. Він може на пряму працювати з конкретними ролями. Такий підхід знижує ризик надмірних дозволів, що буває причиною появи збоїв в системі. При використанні ролей за моделлю RBAC у хмарних сервісах, його мета зазвичай полягає у визначенні доступних API, дозволених та заблокованих операцій.

Модель контролю доступу ABAC визначає який набір прав отримує користувач за допомогою певних властивостей користувача, середовища або іноді самої дії. Така модель немає фіксованих ролей, як RBAC. Вона дивиться на набір характеристик користувача і порівнює їх з політикою доступу. Можна отримати динамічну схему керування, яка добре підходить для керування складними (контекстно залежними) системами. Атрибути можуть містити в собі посаду, номер відділу чи рівень довіри до користувача. Наприклад атрибути середовища можуть описувати тип пристрою, часовий пояс чи місцезнаходження. На основі подібних даних формується набір правил, який і визначає доступні операції в системі. Потенційно така модель надає більший контроль над

дозволом для авторизації. Одна з переваг ABAC в тому, що він має більше можливостей до масштабування. Маючи більше можливих варіантів для налаштування авторизації. Він ймовірно краще підходить під складні бізнес-процеси. Зазвичай у компаніях з великою кількістю працівників, де необхідно багато ролей під різні процеси, ABAC дозволяє краще продумати комплексну модель доступу до елементів системи. Така модель дає можливість більш ефективно реалізовувати IAM політики процесів, щоб запобігати небажаному доступу до чутливих даних та керувати робочим процесом.

IAM (Identity and Access Management) — це комплексна система політик та процесів, яка націлена на покращення управління користувачами в інформаційній системі. Вона відповідає за керування цифровими ідентичностями користувачів та їхнім доступом до конкретних елементів системи. Фокусуючись на надані тільки тих прав для впливу на систему, які необхідні для виконання визначених обов'язків. У IAM відбувається різні процеси для менеджменту облікових записів. Наприклад такі процеси можуть адмініструватись за допомогою IAM: керування ролями, авторизація, політики доступу, моніторинг активності тощо. Система може визначати користувачів, у яких є доступ до входу та які саме ресурси доступні конкретно для нього. Така система політик може допомогти з організацією управління доступом до систем та зменшує ризики помилок від людського фактору. Тому IAM має змогу покращити безпеку даних, особливо у великих компаніях, і дає можливість більш ефективно керувати доступом до сервісів.

Авторизація створює ще один рівень для захисту системи. Вона може гарантувати надання доступу користувачу лише до тих даних, які йому призначені. При цьому доступ у інших місцях системи все ще буде закритим. Таким чином утворюється структурована та більш безпечна

модель управління даними у веб-додатках і інших типах інформаційних систем.

1.2 Парольна аутентифікація та її недоліки

Пароль — це метод аутентифікації за допомогою таємної комбінація символів, який використовується для підтвердження особи користувача під час автентифікації і подальшої авторизації.

Пароль був першим впровадженим методом аутентифікації і таким залишається навіть на сьогоднішній час. На це є свої причини, як наприклад простота в реалізації цього способу. Така аутентифікація передбачає перевірку користувача після введення таємної комбінації символів (зазвичай чисел). Паролі найпростіший у впровадженні спосіб захисту системи, але вони мають достатню кількість недоліків в плані безпеки. Паролі вразливі до фішингу, брутфорс-атак та інших різновидів кіберзлочинів. Надійність знижує повторне використання однакових паролів для різних систем та слабкі комбінації символів при його створенні. З розвитком технології почали з'являтися нові методи захисту систем, які намагались перевершити паролі в надійності. Оскільки навіть складні паролі часто далеко не найбільш надійний спосіб захисту даних.

До розповсюджених прикладів вразливостей та проблем з паролями належать:

1. Потреба у складних комбінаціях;
2. Загроза фішингу;
3. Проблеми з UX;
4. Загрози брутфорс-атак.

Найбільш розповсюдженою проблемою були й залишаються слабкі комбінації. Ця слабкість парольного методу аутентифікації часто пов'язана з іншими загрозами безпеки. Користувачі зазвичай обирали прості або передбачувані паролі, що значно знижувало захист від

несанкціонованого доступу до даних. З часом реєстрація паролю почала вимагати більш складні комбінації символів. В певній мірі це підвищило надійність паролів, але із-за цього користувачу приходиться запам'ятовувати велику кількість складних паролів. При викраденні паролю (який не унікальний для кожного окремого сервісу), якщо навіть складний пароль використовувався, з'являється ризик витоку даних одразу з декількох сервісів. Використання складних комбінацій може призвести до проблем на рівні людського фактору. Можливий сценарій при яком користувач помилиться при вводі такого пароля, що може призвести до блокування на певний період часу чи в гірших випадках назавжди. Одна з функцій такого блокування ймовірно по задуму і є протидія брутфорс-атакам. Щоб зловмисник не мав багато спроб для підбору пароля. Проблема у тому, що механізм працює не тільки проти атак зловмисників, а і проти самих користувачів. Він може забути пароль чи переплутати певні символи, що теж призведе до спрацювання цього захисного механізму. Можливі випадки при яких зловмисники навмисно блокують акаунти користувачів і не мають першочергового наміру вкрадати дані. Наприклад для перевантаження запитами серверу, що робить систему вразливою до інших видів атак.

Під час фішингу відбувається спроба викрадення конфіденційних даних за допомогою соціальної інженерії. Зазвичай зловмисник намагається видати себе за довірену особу. Для цього можуть використовуватися підроблені сайти чи повідомлення на електронну пошту, в які зловмисник намагається видати себе за співробітника справжньої організації. Призводячи до випадків коли викрадають не тільки паролі, а і іншу чутливу інформацію. Ціллю такої атаки можуть бути банківські дані, паролі чи інша персональна інформація. Користувача можуть переконати передати чутливі дані добровільно, маніпулюючи його рішеннями. Системи, які основуються на парольному

захисті, зазвичай часто мають досить високі ризики бути не ефективними проти такого виду кіберзлочину.

При брутфорс-атаці зловмисник перебирає всі можливі комбінації символів для підбору правильного пароля, з подальшим вторгненням. Така атака не залежить від користувача, оскільки зловмисник не взаємодіє з ним на пряму. Існують різні підходи до такої атики, де деякі навіть використовують самописне програмне забезпечення. Більш прості паролі особливо вразливі до такого різновиду нападу. Якщо сервер не має обмежень на кількість спроб чи не застосовує механізми блокування облікових записів, то це може призвести до втрати доступу до несанкціонованої авторизації. Захист від брутфорсу зазвичай включає ліміти спроб входу та паузи між запитами на сервер.

UX (User Experience) — описує загальний досвід користувача від взаємодії з системою чи продуктом. Термін може включати зручність від взаємодії, зручність у використанні, швидкість та загальну ефективність роботи з точки зору користувача. Добре впроваджений UX повинен робити взаємодію простою і не перевантажувати користувача.

В аутентифікації за допомогою паролів UX зазвичай має певні проблеми. Хоча паролі і прості у використанні, але з точки зору безпеки накладають на користувача багато відповідальності. Іноді користувачі ігнорують вимоги до складних паролів заради зручності і комфорту у використанні, що призводить до зниження захисту від потенційних загроз. Незручності можуть створювати ситуації коли потрібно оновити пароль (був втрачений), або якщо для оновлення паролю потрібно залучати інших людей. Подібні випадки можуть значно знизити якість досвіду взаємодії з системою або червісом.

Багатофакторна аутентифікація (MFA) — це підхід до авторизації, при якому для підтвердження особи використовується більше одного фактора перевірки. Коли один фактор може бути скомпрометований, то

інші створює додатковий бар'єр захисту. Таке поєднання декількох факторів створює додатковий рівень безпеки. Багатофакторна аутентифікація покладається не тільки на пароль, а і на другий фактор. Ним можуть бути наприклад токени, апаратні ключі тощо. Комбінація декількох факторів допомагає попередити несанкціонований вхід, але не гарантує абсолютну безпеку. Багатофакторна аутентифікація може поєднуватись з безпарольними технологіями, де виступаючи додатковим захистом для системи. Тому багатофакторна аутентифікація гарний спосіб підвищити безпеку користувача. Хоча вона потребує додаткових витрат для підтримання такої архітектури задля ефективного функціонування.

1.3 Розвиток безпарольних технологій

Безпарольна аутентифікація — це метод аутентифікації, який дозволяє отримати доступ до системи без введення пароля. До розповсюджених безпарольних методів відносятться: біометричні дані, апаратні токени та одноразові коди.

Існує інше розгалуження напрямку аутентифікації, яке не покладається на паролі. Безпарольні технології спирається на криптографічні ключі, біометричні дані чи одноразові токени та інші методи. Повністю він не виключає використання паролю на певному етапі, але не спирається на паролі, як на основний етап захисту даних. Біометричні дані у безпарольних технологіях відносяться до унікальних фізіологічних даних користувача. До них належать відбитки пальців, розпізнавання обличчя, голосу тощо. Такі дані здебільшого важко підробити. Натомість апаратні токени представляють з себе фізичні пристрої, такі як флеш накопичувачі. Вони зберігають приватний ключ у захищеному середовищі і під час запиту на аутентифікацію посилають запит на сервер. Тому токен не взаємодіє з сторонніми чинниками. Система отримує цифровий підпис, який майже неможливо підробити

без доступу до самого пристрою. Вони виступають, як високозахищенні аутентифікатори. Часто він під'єднується до пристрою через доступні методи (usb порти, Bluetooth), після чого потребує дотику для підтвердження.

Одноразові коди зазвичай потребують номер телефону або електронну пошту для отримання код (аналог паролю), щоб підтвердити вхід в систему. Такий сервіс зазвичай не потребує багато витрат для обслуговування. Оскільки проблеми з відновленням доступу чи обслуговування паролів не вимагають багато ресурсів. Вони досить доступні, бо їх можна отримати на будь-який пристрій, який має доступ у мережу інтернет і підтримує браузер. Одноразові коди мають вразливості такі як: перехоплення повідомлення чи компрометація електронної пошти. Тому одноразові коди хоч і зручний в реалізації метод, але не найбільш надійний серед безпарольних рішень.

Серед переваг безпарольної аутентифікації зазвичай виділяють наступні з точки зору досвіду у використанні:

- Знижується ризик фішингу. Оскільки відсутність покладання відповідальності за безпеку на пароль знижує ризик його перехоплення.
- Зазвичай кожен ключ чи токен унікальний, тому немає потреби у запамятовуванні та створенні складних паролів. Це може зменшити ймовірність помилок при вході та підвищує загальний досвід аутентифікації.

На даний момент не існує ідеального методу аутентифікації. Хоч безпарольні технології і перевершили паролі з часом. Вони все ще потребують вдосконалення та пошуку ще кращих рішень. Можливо така безпарольная технологія, як WebAuthn може краще себе показати порівняно з іншими безпарольними методами.

1.4 Стандарти FIDO2, CTAP і WebAuthn

WebAuthn один із відносно нових стандартів сучасної безпарольної автентифікації. Він об’єднує використовує криптографічні ключі для забезпечення надійного та зручного входу в систему. Загалом криптографічні ключі поступово набирають популярність у певних сферах діяльності людини. Для повноцінного функціонування WebAuthn потребує інші протоколи, такі як FIDO2 та CTAP.

FIDO2 — це сучасний відкритий стандарт, який створений для безпарольної аутентифікації у веб-сервісах. Він поєднує два протоколи WebAuthn та CTAP. Разом вони дозволяють користувачу входити до веб-сервісів за допомогою криптографічних ключів, біометрії чи апаратних токенів.

FIDO2 переносить інформацію ключів для аутентифікації з сервера у захищене середовище. Таким середовищем може виступати Secure Enclave, TPM або просто апаратний ключ. Криптографічний підпис прив'язаний до конкретного веб-сайту чи додатку. Тому ризик фішингу може зменшуватись. Цей стандарт підтримує вбудовані у смартфони чи ноутбуки автентифікатори (Windows Hello, Touch ID, Android Biometrics) і зовнішні ключі YubiKey. Автентифікація стає швидшою, бо для неї потрібно просто достатньо доторкнутись до датчика, прикласти ключ або підтвердити дію на смартфоні.

Вбудована в операційну систему Windows технологія біометричної безпарольної аутентификации Windows Hello дозволяє користувачам швидко заходити в систему. Вона використовує такі методи, як розпізнавання обличчя, відбитків пальців або паролі. Вона прив’язується до конкретного пристрою, що знижує можливість використання даних на сторонніх системах. Для WebAuthn Windows Hello може виступати як аутентифікатор платформи, який буде інтегрований в операційну систему. Він зберігає криптографічні ключі на пристрої та підтримує створення publicKeyCredential. Використання Windows Hello у поєднанні

з TLS/HTTPS та серверною валідацією дозволяє впроваджувати надійн системи доступу. Знижуючи ризик перехоплення облікових даних. Також підвищує загальний рівень безпеки при використанні.

Система автентифікації Android Biometrics інтегрована в операційну систему Android. Вона дозволяє використовувати відбитки пальців, розпізнавання обличчя та інші біометричні дані для забезпечення подальшого доступу. Біометричні дані зберігаються в захищеному сховищі (Trusted Execution Environment) і не передаються на сервери.

Біометрична технологія Touch ID розроблена для роботи на операційній системі IOS. Ця технологія використовує відбиток пальця для безпечного доступу до пристроїв та онлайн-сервісів. Вона впроваджена в апаратній частині пристрою. Можливе зберігання криптографічних ключі у захищеному сховищі Secure Enclave. Вона не передає відбитки пальців на зовнішні сервери, що підвищує рівень безпеки даних та зберігає приватность.

Зовнішній апаратний аутентифікатор YubiKey забезпечує безпечний доступ за допомогою криптографічних ключів. Пристрій підтримує стандарти WebAuthn та FIDO2. Він дозволяє здійснювати аутентифікацію за допомогою дотику до пристрою аутентификатора чи введення паролю. YubiKey теж генерує та зберігає приватні ключі локально. Він підтримує різні інтерфейси, такі як USB різних типів, NFC та Lightning. Що забезпечує широке охоплення платформ і пристроїв. Загалом YubiKey є досить популярним рішенням.

CTAP (Client To Authenticator Protocol) — це протокол взаємодії між пристроєм клієнта (наприклад браузером) та аутентификатором (зовнішнім або вбудованим). Він забезпечує захищений канал для виконання криптографічних операцій під час безпарольної аутентификации. Аутентифікатор з CTAP може генерувати пари

криптографічних ключів, створювати підписані відповіді для серверів та виконувати атестацію не розкриваючи приватний ключ. У межах FIDO2, на даний момент, існують два основні варіанти протоколу.

- CTAP1 — версія орієнтована на двофакторну аутентифікацію з використанням апаратних ключів. Вона забезпечує простий і дієвий механізм підтвердження входу.
- CTAP2 — розширений варіант, який підтримує повноцінну безпарольну автентифікацію. Він працює разом із WebAuthn та може обробляти криптографічниі даниі.

Протокол дозволяє автентифікаторам під’єднуватися через, певні канали. Також можливе з'єднання через вбудовані модулі TPM, Secure Enclave чи Android Keystore. Тож цей протокол досить універсальний і придатний для широкого спектра пристроїв. Розповсюджені інтерфейси USB, BLE (Bluetooth Low Energy) та NFC використовуються апаратними автентифікаторами для взаємодії з пристроями користувача. Вони мають свої особливості та сфери застосування. Вони повинні забезпечувати передачу даних між автентифікатором і пристроєм.

Підключення через USB є поширеним методом, який забезпечує досить стабільну комунікацію. Пристрої, що підключаються через USB, зазвичай використовуються на стаціонарних комп’ютерах та ноутбуках. Таке підключення створює низьку затримку у передачі даних. Також воно сумісне з різними операційними системами. Бездротово взаємодіяти автентифікаторам з пристроями дозволяє BLE. Зазвичай він діє на відстані до кількох метрів. BLE споживає мінімальну енергію, що робить його гарним рішенням в плані економії ресурсів. Зазвичай BLE використовується коли коли фізичне підключення через USB неможливе. NFC протокол використовується для швидкої та інтуїтивної автентифікації. Для цього користувач повинен піднести автентифікатор до пристрою, який повинен прочитати ключ і відтворити безпечний

обмін даними. Для мобільних пристроїв зазвичай використовується NFC. Такий метод забезпечує особливо високу швидкість та зручність, порівняно з іншими двома способами. Сукупність цих технологій у сучасних апаратних аутентифікаторах дозволяє обирати необхідний спосіб аутентифікації.

Апаратний модуль безпеки TPM (Trusted Platform Module) забезпечує надійне зберігання криптографічних ключів. Він інтегрований у пристрої та відповідає за виконання різних криптографічних операцій. TPM створює апаратне середовище, де ключі зберігаються, що ускладнює їх компрометацію. TPM використовується як платформа для генерації приватних ключів та їх зберігання. Модуль підтримує криптографічні алгоритми та запобігає експортованню ключів, навіть якщо операційна система або додатки були атаковані. Також TPM дозволяє реалізувати посвідчення ключів. Тобто підтвердження що ключі були створені на надійному пристрої. Завдяки TPM системи з WebAuthn можуть покладатися на захист даних, який підкріплений апаратно для надійної безпарольної аутентификации.

Ізольований апаратний підсистемний модуль Secure Enclave використовується в пристроях IOS для забезпечення високого рівня безпеки обробки та зберігання криптографічних ключів і чутливої інформації. Secure Enclave працює окремо від основного процесора, де ключі не покидають ізольовану пам’ять. Це робить їх недоступними для операційної системи та сторонніх додатків. Завдяки ізоляції ключі залишаються захищеними від зловмисників. Інше розповсюджене рішення для мобільних пристроїв - це Android Keystore. Він відноситься до програмно захищених механізмів на платформі Android. Його задача керувати та взаємодіяти з криптографічними ключами. Він ізолює приватні ключі від основного програмного середовища так само, як і Enclave.

Застосуванні CTAP може забезпечити:

- Захищене делегування операцій аутентификатору. При цьому не передаючи приватні ключі.
- Сумісність між різними платформами та аутентификаторами.
- Взаємодію користувача лише зі своїм пристроєм (смартфоном, токеном чи сенсором) для авторизації.

WebAuthn (Web Authentication) — це веб-стандарт аетентифікації, який дозволяє входити в онлайн-сервіси без пароля. Для цього він використовує криптографічні ключі, апаратні токени або біометрію. Він входить до складу FIDO2 та забезпечує безпечний механізм підтвердження особи. WebAuthn не передає криптографічні ключі через мережу.

WebAuthn зберігає приватний ключ на пристрої користувача (наприклад у вигляді токена). Натомість публічний ключ надається серверу в момент авторизації чи реєстрації. Сервер надсилає унікальний сигнал коли користувач намагається увійти, а пристрій створює криптографічний підпис з приватним ключем. Це ускладнює перехоплення трафіку зловмиснику, бо підпис прив’язаний до певного домену і деактивується через певний час. Стандарт WebAuthn підтримує різні моделі автентифікаторів. Це можуть бути зовнішні токени або вбудовані, як Windows Hello чи Face ID. Через відбиток пальця чи пароль WebAuthn може локально перевіряти користувача.

1.5 Порівняння розповсюджених методів парольної та безпарольної аутентифікації

Були розглянуті парольні та безпарольні методи аутентифікації. Всі вони мають свої переваги та недоліки. У рамках розділу було розглянуто такі розповсюджені методи аутентифікації:

1. Парольна аутентифікація є досить поширеним методом аутентифікації завдяки простоті в реалізації. Основується на

перевірці збігу введеного користувачем пароля із збереженим хешем на сервері.

2. Багатофакторна автентифікація підвищує рівень безпеки, комбінуючи декілька факторів. Багатофакторна автентифікація значно зменшує ризик компрометації облікового запису. В тому числі у випадку успішної атаки на один з факторів.
3. Токени та сертифікати використовуються для аутентифікації без необхідності введення пароля. Замість цього використовують криптографічні діні. Керування такими токенами та сертифікатами може бути складним. Особливо при необхідності обслуговування великих обсягів інформації для оновлення чи виправлення даних.
4. Криптографічні ключі для аутентифікації використовують підписання повідомлень за допомогою цифрового підпису. Сам ключ зберігається в безпечному середовищі, наприклад у TPM або Secure Enclave. Потребує підтримки відповідних аутентифікаторів. Іноді може бути несумісним з деякими системами або платформами.
5. Біометрична автентифікація базується на унікальних фізичних характеристиках користувача. Це можуть бути відбитки пальців або риси обличчя. Забезпечує швидкий безпарольний підхід та ускладнює підробку аутентифікаційних даних. Має проблеми з приватністю та конфіденційністю. Також потреба у спеціальному обладнанні для збору та оброблення біометричних даних.
6. Апаратні токени оскільки ключі зберігаються на фізичному носії. Тому вони гарантують не можуть бути перехоплені через мережу інтернет. Потребують використання і зберігання фізичного пристрою. Може призвести до загублення чи пошкодження токена (ключа).

7. Одноразові коди простий та доступний метод. Він часто використовуються у SMS або через апаратні генератори кодів. Вони забезпечують додатковий рівень безпеки в поєднанні з паролем. Проте вони можуть бути перехоплені, що знижує їхню надійність. Якщо порівнювати з криптографічними ключами або апаратними токенами. Особливо у корпоративних середовищах з високими вимогами до безпеки.

Враховуючи розглянуті аспекти різних методів аутентифікації можна зробити порівняльну таблицю. Щоб визначити переваги та недоліки різних методів аутентифікації. Враховуючи як парольні, так і безпарольні рішення.

Метод аутентифікації	Переваги	Недоліки
Аутентифікація через електронну пошту	Має просту реалізацію, не потребує додаткової архітектури для впровадження.	Не гарантує високий рівень безпеки. Вразлива до фішингу. Найбільш вразливий метод безпарольної аутентифікації.
Багатофакторна автентифікація (MFA)	Забезпечує підвищений рівень безпеки, зменшує ризик компрометації облікового запису.	Потребує додаткової інфраструктури. Може погіршувати досвід користувача, оскільки вимагає введення додаткових кодів або підтверджень.
Біометрична автентифікація	Проста у використанні. Ускладнює підробку ключів для аутентифікації.	Вимагає спеціального обладнання та інфраструктури. Наприклад сканери відбитків пальців чи камери для розпізнавання обличчя.
Апаратні токени	Високий рівень	Потребують

	безпеки, стійкість до фішингу. Ключі фізично захищені від атак поки не контактують з пристроєм.	фізичного носія. Можливий ризик втрати або пошкодження токена.
Одноразові коди (OTP)	Простота використання. Доступні великому обсягу користувачів. Можуть бути інтегровані з мобільними додатками.	Мають ризик перехоплення під час передачі. Менша надійність у порівнянні з іншими методами.

Таб. 1.1 – Порівняння переваг та недоліків основних методів безпарольної аутентифікації

У порівнянні з іншими методами, апаратні токени та біометрична аутентифікація поєднує надійність із зручністю користувача, забезпечуючи баланс між безпекою та UX. Інтеграція цих методів у веб-додатки може дозволити зменшити кількість атак, пов’язаних із паролями, і створити більш стійку автентифікаційну систему.

Висновок до розділу 1

Було розглянуто теоретичні основи аутентифікації у веб-додатках та її розвиток. Визначено ключові поняття аутентифікації та авторизації та підкреслено їхнє значення для забезпечення інформаційної безпеки та управління доступом до ресурсів системи. Авторизація забезпечує контроль над рівнями доступу користувачів, а моделі RBAC та ABAC дозволяють ефективно впроваджувати політики доступу з урахуванням ролей і атрибутів користувачів.

Розглянуто недоліки традиційної парольної аутентифікації, серед яких фішинг, брутфорс-атаки, проблеми UX та необхідність складних комбінацій символів. Ці висновки приводять до того що паролі не

завжди є надійним та зручним рішенням. Визначено, що багатофакторна аутентифікація підвищує безпеку, але потребує додаткових ресурсів та може ускладнювати користувацький досвід.

Окремо були розглянуті бепарольні технології, які використовують криптографічні ключі, біометрію, апаратні токени та одноразові коди. Такі методи забезпечують більш високий зручність для користувачів, знижуючи ризик фішингу та проблем з управлінням паролями. Стандарти FIDO2, CTAP і WebAuthn демонструють перспективність сучасних рішень для веб-автентифікації, поєднуючи криптографічний захист із простотою використання.

Порівняння різних методів аутентифікації показало, що WebAuthn та інші безпарольні підходи забезпечують оптимальний баланс між безпекою та зручністю користувача. Тому можуть бути перспективним напрямком розвитку сучасних веб-додатків.

2 АРХІТЕКТУРА ТА ПРИНЦИПИ РОБОТИ ТЕХНОЛОГІЇ WEBAUTHN

2.1 Загальна характеристика та важливі поняття для розуміння принципів роботи WebAuthn

Стандарт для безпечної аутентифікації WebAuthn не потребує використання паролів. Він визначає API для браузерів та платформ, щоб веб-додатки могли створювати та перевіряти криптографічні ключі. WebAuthn ставить перед собою цілі забезпечення надійної та зручної аутентифікації. Гарний баланс цих двох аспектів зможе покращити загальний UX.

API (Application Programming Interface) — це набір механізмів, які дозволяють різним програмним системам взаємодіяти між собою. API використовується для обміну токенами. Він може використовуватись для перевірки прав доступу або взаємодії з апаратними аутентифікаторами. Сервер може викликати API для перевірки підпису WebAuthn. Такі системи можуть бути модульними і взаємодіяти один з одним без потреби у прямій інтеграції на рівні коду. Автентифікатор працює з браузером через WebAuthn API та протокол CTAP. Забезпечуючи захищену генерацію ключів і підписів. API зазвичай виступає у вигляді інтерфейсів REST, GraphQL або gRPC

Програмне середовище, яке надає готову структуру та набір інструментів для розробки програмних додатків відноситься до фреймворків. Вони дозволяють швидко впроваджувати механізми безпарольної аутентификації, забезпечуючи обробку “publicKeyCredential” та інших даних. З подальшою заємодією з сервером через протоколи (gRPC, REST, GraphQL тощо). Використання фреймворку зменшує ймовірність помилок. Також прискорює розробку та полегшує підтримку коду завдяки вже готовим шаблонам. Фреймворки у WebAuthn можуть включати модулі для генерації

“challenge”. Мають інтеграції з базою даних для управління сесіями. Забезпечують механізми безпечної передачі даних через TLS/HTTPS. В тому числі готові інтерфейси для взаємодії з JavaScript API на клієнтській стороні. Використання фреймворків дозволяє створювати масштабовані системи аутентифікації, що зменшує час розробки.

REST представляє архітектурний стиль для створення веб-сервісів. Він орієнтований на використання стандартних протоколів. Він дозволяє організувати обмін даними через чітко визначені ресурси і HTTP. REST не прив’язаний до конкретного формату даних. У WebAuthn зазвичай використовуються JSON структури для передачі payload між клієнтом і сервером. Використання REST може забезпечити сумісність між різними серверними рішеннями. Разом з TLS/HTTPS він забезпечує безпечну передачу даних. Його додаткові властивості дозволяють серверу обробляти кожен запит незалежно, що підвищує відмовостійкість системи. Застосування REST у WebAuthn може дапоготи утворити гнучку взаємодію між компонентами системи.

Середовище та мова запитів GraphQL дозволяє контролювати, які дані надходять клієнтам. У той час, як REST надає стандартизовані відповіді. GraphQL в поєднанні з WebAuthn може використовуватися для обробки даних автентифікації. Такими даними можуть бути “publicKeyCredential” чи “assertion”. Користувачу буде легше визначити, які поля потрібні для виконання необхідної йому операції. Наприклад для реєстрації чи для в ходу у сервіс. GraphQL надає певні можливості для зручної обробки даних на сервері. Можна будувати більш складну централізовану логіку для аутентифікації. З TLS/HTTPS надається більш безпечна передача даних. Типізована структура дозволяє виявляти помилки ще на етапі формування запитів, що підвищує надійність системи. GraphQL може бути інтегрованим у сучасні веб-додатки. Він

забезпечує взаємодію між елементами системи, яку легше контролювати.

Для зберігання та обміну даними можна використовувати Protocol Buffers (Protobuf). Він замінює іноді громіздкі формати на кшталт XML чи JSON. Він представляється, як гнучкий не залежний від платформи механізм серіалізації структурованих даних. В нього входять можливості забезпечення компактного зберігання та обміну даними між різними системами. Також він підтримує різні мови програмування та сервери. Концепція Protocol Buffers полягає в описі структури даних за допомогою спеціальної схеми. Для цього він використовує свій формат даних. Файл визначає повідомлення з полями різних типів. Кожне поле має унікальний ідентифікатор, який використовується під час серіалізації. Після опису структури його можна скомпілювати у код для конкретної мови програмування. Такий спосіб дозволяє легко створювати та читати повідомлення. Protocol Buffers має ефективний для зберіганні даних з ціллю швидкої серіалізації та десеріалізації. Обсяг переданих даних зменшується завдяки бінарному формату. Це дозволяє працювати з системами з обмеженою пропускною здатністю. В Protobuf можна додавати нові поля без порушення сумісності зі старими версіями програми. Така можливість полегшує роботу при розробці довгострокових проектів.

Технологія Protocol Buffers зазвичай застосовуються у внутрішніх сервісах компаній для яких важлива швидкість обміну даних. Також зустрічається у мобільних додатках, де економія трафіку має сенс. Protobuf інтегрується з сучасними технологіями RPC та gRPC, що може дозволити організувати продуктивну взаємодію між сервісами.

Високопродуктивний фреймворк gRPC використовує протокол HTTP/2 та бінарний формат передачі даних Protocol Buffers. Також він підходить для віддаленого виклику процедур (Remote Procedure Call).

Ціль RPC в тому, щоб приховати складні мережеві взаємодії і забезпечити простий інтерфейс виклику процедур. Що робить мережевий рівень стає майже непомітним. Таким чином під час виклику RPC клієнтська програма формує запит. Цей запит містить ім’я віддаленої процедури та необхідні аргументи. Потім він передається через мережу до сервера, який обробляє його. Виконуючи відповідну процедуру та повертає результат назад. Цей процес відбувається у напівавтоматичному режимі. В той час як розробник взаємодіє з процедурою як із локальною функцією. Загалом RPC спрощує створення розподілених систем. Він дозволяє розділити логіку між сервісами без необхідності управління протоколами або мережевою інфраструктурою. Мережеві деталі інкапсулюються в моделях, які генеруються на стороні клієнта і сервера. Потім вони відповідають за перетворення викликів у мережеві повідомлення. З подальшою зворотною обробку отриманих даних. Для ефективної взаємодії між клієнтськими додатками та серверною частиною gRPC може застосовуватися для обміну даними аутентифікації та метаданих аутентифікаторів. Використання бінарного формату дозволяє мінімізувати обсяг переданих даних. Тому він моє змогу забезпечити швидку передачу, що корисно для масштабних систем із високим навантаженням. Модель RPC часто є основою для більш сучасних технологій. Наприклад gRPC, яка наслідує концепти RPC, додаючи підтримку контрактно орієнтованих API. Крім цього дозволяє реалізовувати високошвидкісні бінарні серіалізації.

На наш час RPC залишається достатньо надійним концептом для розуміння архітектури розподілених систем. Основуючи основні принципи побудови взаємодії між сервісами в сучасному програмному забезпеченні. строго типізовані інтерфейси можуть підтримуватись gRPC, що дозволяє на етапі компіляції перевіряти коректність запитів та відповідей. Такий підхід знижує ймовірність помилок у процесі

аутентифікації. Завдяки інтеграції з TLS/HTTPS забезпечується безпечна передача даних. Використання gRPC для WebAuthn дозволяє створювати захищені та масштабовані системи аутентифікації. Вони підходять для корпоративних і хмарних сервісів.

Стандарт FIDO2, він включає два основні компоненти:

- WebAuthn, який допомагає з взаємодією між веб-додатком та аутентифікатором користувача.
- CTAP, що забезпечує комунікацію між клієнтським пристроєм та аутентифікатором (зовнішнім або вбудованим).

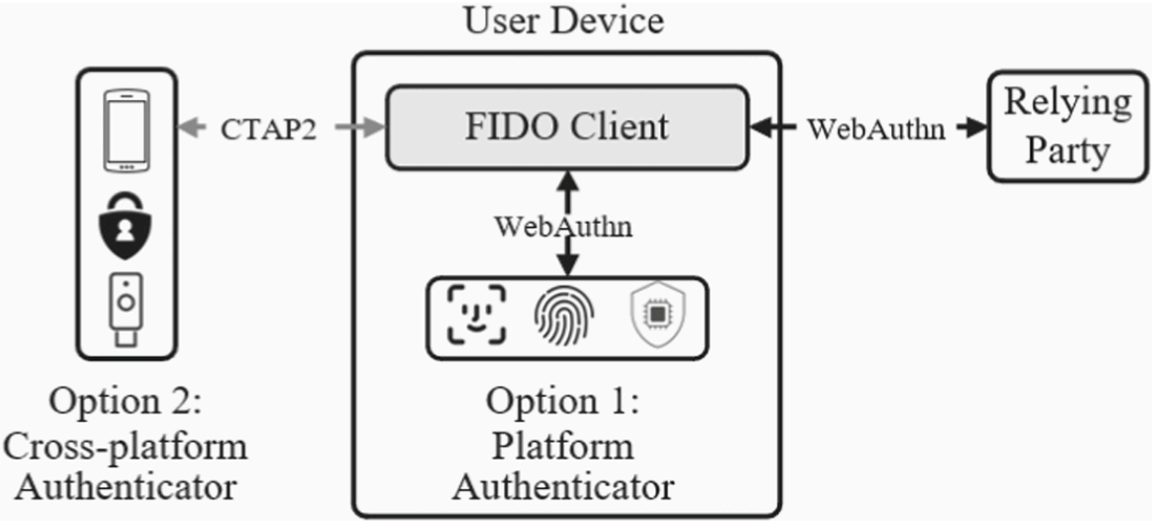


Рис. 2.1 – Архітектура FIDO2.

Система FIDO2 дозволяє використовувати різноманітні типи автентифікаторів, що надає широку сумісність та гнучкість. WebAuthn забезпечує сильну аутентифікацію, уникаючи проблем традиційних паролів, таких як фішинг чи інші подібні.

Походження (Origin) — це поняття, яке визначає межі того, які ресурси та скрипти можуть взаємодіяти між собою в браузері. Воно складається з трьох елементів: схеми, домену та порту. Якщо всі три

збігаються, то браузер вважає два ресурси частиною одного походження. Якщо один із параметрів відрізняється, то це вже іншие походження. Такий випадок підпадає під обмеження політики SOP (Same-Origin Policy). Origin визначає, які веб-сторінки мають право взаємодіяти між собою. Браузер зазвичай суворо дотримується цього правила, щоб запобігати атакам. Наприклад доступу до локальних сховищ або перехоплення контенту з інших сайтів. Два практично ідентичних домена все ще будуть мати різне походження, бо відрізняється певна частина домену. Це важливо при побудові складних веб-додатків, де багато компонентів працюють під різними доменами.

Для WebAuthn та FIDO2, термін походження набуває ще одного значення. Приватний ключ апаратного автентифікатора прив’язується до походження під час реєстрації. Тому навіть якщо зловмисник створить сайт, який буде практично ідентичний (інший домен чи протокол) з оригінальним, то аутентифікатор не підтвердить на ньому спробу входу. Прив’язка до походження майже повністю усуває загрозу фішингу, оскільки приватний ключ просто не буде працювати в іншому середовищі. Походження визначає межу довіри в моделі безпеки браузера. Він допомагає визначити де закінчується зона контролю однієї веб-програми та починається інша. Без цього механізму взаємодія між сайтами була би неконтрольована і небезпечна. Оскільки тоді будь-який ресурс міг би отримати доступ до даних користувача, які зберігаються у його браузері. Завдяки сходженню веб-сервіс зберігає ізольованість компонентів, що забезпечує захищене виконання скриптів. Також він допомагає будувати складні системи аутентифікації та авторизації. В яких присутня перевірка контексту, в якому виконується той чи інший запит. Крім цього походження також є частиною CORS (Cross-Origin Resource Sharing). Механізм CORS дозволяє або забороняє доступ до ресурсів між різними доменами. Сервер визначає за допомогою

заголовків CORS, чи дозволяти стороннім походженням виконувати запити.

CORS (Cross-Origin Resource Sharing) — це механізм, який дає веб-серверам можливість контролювати, які зовнішні походження (Origins) можуть отримувати доступ до їхніх ресурсів. Він розширює суворі правила SOP (Same-Origin Policy). При цьому не порушуючи їх, а доповнюючи керованими винятками. Коли сервер підтверджує довіру до джерела, то CORS дозволяє браузеру виконувати запити між різними доменами. Все інше, ще до передачі конфіденційних даних, браузер блокує автоматично.

Принцип CORS основується на обміні заголовків HTTP. Коли сторінка з одного походження запитує ресурс на іншому, то браузер перевіряє сервер на відповідь з заголовком Access-Control-Allow-Origin. Якщо дозволу немає, то дані не передаються через JavaScript, навіть якщо запит виконано правильно. Відповідальність за довіру лежить на сервері, а браузер лише дотримується інструкцій. Одним із аспектів CORS є запити Preflight, які браузер відправляє перед основним. В випадках, якщо той може бути потенційно небезпечним. Такі запити використовують метод OPTIONS і перевіряють, чи сервер допускає певні типи вмісту. Тож браузер не надішле критичний чи невідворотний запит без чіткого дозволу. Це дозволяє попередити атаки, коли зловмисники намагаються змусити браузер користувача виконати небажані дії на сторонньому веб-сервісі. Також CORS впливає на обробку токенів та інших облікових даних. Для того щоб їх передати через між доменний запит, сервер повинен не лише дозволити Origin. Крім цього він повинен вказати “Access-Control-Allow-Credentials: true”. Браузер же вимагатиме точне, а не універсальне походження у відповідь. Це робить пересилання авторизаційних даних більш контрольованим, що знижує ризик несанкціонованого доступу. У масштабних

веб-архітектурах CORS представляє з себе інструмент управління довірою між мікросервісами, фронтендом та API. Особливо якщо елементи системи працюють на різних доменах. Він дозволяє розмежувати доступ та більш чітко визначити, які частини системи можуть взаємодіяти між собої. В тому числі може спростити інтеграцію з зовнішніми платформами.

Same-Origin Policy (SOP) — визначає, які веб-ресурси можуть взаємодіяти між собою. Це ймовірно один із найбільш важливих елементів безпеки браузера. Він запобігає неконтрольованому доступу між сайтами та захищає користувача від крадіжки даних чи несанкціонованого виконання скриптів.

Політика базується на порівнянні трьох параметрів: схеми, домену, порту. Якщо всі три збігаються, то ресурси вважаються частиною одного походження. Після чого вони можуть виконувати взаємодії без обмежень. Цей принцип важливий для захисту браузерних середовищ, оскільки без нього будь-який скрипт з будь-якого сайту зміг би читати інформацію з іншого або отримувати доступ до локального сховища. Завдяки SOP забезпечується повна ізоляція між незалежними веб-додатками. Також ця політика допомагає захищати конфіденційні дані. Коли користувач авторизується на певному сайті, то браузер зберігає маркери доступу та інші приховані дані. SOP запобігає випадкам коли ці дані стануть доступними стороннім сторінкам, які навмисно або випадково можуть бути вбудовані через iframe чи перенаправлення. Ізоляція таких даних не дає змоги зловмисникам виконати типові атаки. Наприклад крадіжку сесії через скрипт із іншого ресурсу. SOP працює разом з іншими механізмами, які дозволяють керовано обходити її обмеження за потреби. В приклад можна привести CORS, що дає серверу можливість дозволяти доступ певним стороннім походженям. SOP і CORS утворюють ієрархію правил, де браузер

спочатку блокує між доменну взаємодію за правилами SOP. Після чого перевіряє, чи дозволив сервер такий доступ через CORS. Така ієрархія дає можливість створювати складні інтеграції, не порушуючи безпеку. У WebAuthn SOP повинен гарантувати, що криптографічні операції виконуються лише в контексті того Origin, для якого аутентифікатор створив криптопару. Якщо зловмисник спробує підробити інтерфейс сайту, то браузер виключить можливість доступу до прив'язаних ключів.

Origin-binding — це принцип безпеки в WebAuthn, який забезпечує жорстку прив’язку аутентифікаційних операцій (створення ключів, підписання челенджу, підтвердження входу) до конкретного походження.

Принцип механізму полягає в тому, що під час створення криптопари ключів автентифікатор не тільки генерує ключі, а і прив’язує їх до походження. Тому приватний ключ, створений для одного веб-сайту, ніколи не буде використаний для іншого. Включаючи випадок, якщо зловмиснику вдалося змусити браузер виконати запит від підробленої сторінки. Коли користувач проходить аутентифікацію, то автентифікатор формує підпис. Цей підпис включає не тільки челендж, але й інформацію про поточне походження. Сервер спочатку перевіряє підпис і потім підтверджує, що операція була ініційована саме в межах довіреного домену. Origin-binding робить фішингові атаки практично марними. Оскільки аутентифікатор відмовиться підписувати челендж для домену, який не відповідає пов'язаному з ключем домену. Це створює модель безпеки, де атаки типу MITM, DNS-спуфінг або підміна сторінки в певній мірі втрачають свою ефективність

Особливість origin-binding також полягає в тому, що вся перевірка здійснюється на рівні браузера та аутентифікатора. Тож немає потреби в передаванні секретної інформації. Браузер контролює контекст, із якого викликаються WebAuthn API, щоб операції виконувалися тільки з безпечного середовища (HTTPS). Це дозволяє уникати сценаріїв підміни

шляху виконання. Тому WebAuthn пропонує досить найнадійний підхід до захисту користувача в веб-додатках. Прив’язка до походження є важливою для побудови безпарольних систем, де логіка будується на криптографічному підтвердженні, яке пов’язане з конкретним веб-додатком.

Сильна аутентифікація значно підвищує рівень безпеки доступу до систем. Вона це робить шляхом використання кількох факторів перевірки (MFA) або криптографічно захищених засобів. Сильна аутентифікація не покладається на знання користувача. Вона має на увазі використання фізичних пристроїв, біометричних даних або одноразові криптографічні ключі. Тому може знизити ризик різних типів атак. Сильна аутентифікація реалізується через використання “publicKeyCredential”. Такий приватний ключ ніколи не покидає автентифікатор. Перевірка відбувається за допомогою криптографічного підпису. Також вона зазвичай інтегрується з SSO та корпоративними політиками безпеки, що дозволяє централізовано контролювати доступ.

SSO (Single Sign-On) — це технологія централізованої аутентифікації. Вона дозволяє синхронізувати доступ одразу до кількох веб-додатків (сервісів) після аутентифікації. Таким чином SSO робить так, щоб користувачу не потрібно було повторно вводити дані для кожного окремого сервісу. Натомість система підтверджує його особу на декількох платформах одночасно.

Робота SSO основується на створенні залежностей між сервісами та системою аутентифікації. Таку систему називають IdP (Identity Provider). Вона відповідає за аутентифікацію користувача і надання інформації про них довіреним сервісам. Ціль IdP в тому, щоб підтвердити особу користувача і передати цю інформацію у вигляді цифрового маркера. Після чого вже сторонні сервіси для надання доступу приймають його для надання доступу. Функціонування IdP

пов’язаний з SSO та протоколом аутентифікації, як наприклад OAuth 2.0. Якщо користувач спробує авторизуватись у веб-додатк, то додаток перенаправить його на IdP для перевірки облікових даних. При успішній аутентифікації IdP зможе згенерувати токен, який буде містити інформацію про користувача. Після чого цей токен буде переданий додатку. В цьому випадку додаток не зберігає паролі користувачів. Оскільки він покладається на IdP, як на достовірне джерело для аутентифікації. Також IdP допомагає з у централізованим управлінні безпекою та доступом. Дозволяючи адміністратору керувати обліковими записами та іншими механізмами безпеки в одному місці. Це дозволяє зберегти цілісність і контроль над усіма сервісами, які використовують IdP. IdP в тому числі дозволяє підвищити зручність користувачів. Завдяки централізованій автентифікації зменшується кількість паролів, які потрібно запам’ятовувати. SSO забезпечує доступ до всіх пов’язаних сервісів після одного входу. Тож IdP може забезпечити надійний обмін інформацією про користувача між різними додатками. При цьому зберігаючи високий рівень безпеки та сумісності з іншими сервісами

Коли користувач намагається увійти у веб-додаток, то сервіс перенаправляє його на IdP. Вже далі, на рівні IdP, відбувається перевірка облікових даних. Після успішної аутентифікації IdP передає сервісу токен, який підтверджує особу. Токен має в собі інформацію про користувача та права доступу. Тож це дозволяє сервісу надати доступ без введення пароля. Технологія SSO може підвищити зручність використання систем для користувача. Вона зменшує кількість паролів, які потрібно запам’ятовувати. Зменшується ризик фішингових атак і повторного використання паролів на сторонніх сервісах. З'являється можливість керування сесіями та токенами. Можна заробити щоб токен дійсний лише протягом встановленого часу, розробити механізми примусового виходу або налаштувати оновлення токена. Можливе

налаштування прав доступу для різних сервісів, щоб користувач отримував доступ лише до дозволеного функціоналу.

Зазвичай SSO використовується у корпоративних середовищах і комерційних веб-сервісах. Оскільки там часто потрібно об’єднати багато елементів системи. При цьому зручність для користувачів має теж значення. Його застосування дозволяє оптимізувати процеси логіна та підвищити UX. Загалом дозволяє забезпечити централізоване управління безпекою та знизити навантаження на інформаційний відділ.

Не менш важливим є застосування сильних аутентифікаторів для захисту чутливих даних, де парольний захист є недостатньою мірою безпеки. Такий аутентифікатор має поєднувати криптографічну стійкість, фізичну присутність користувача та стандартизовані протоколи.

Існують наступні типи аутентифікаторів:

- USB-аутентифікатор
- NFC аутентифікатор
- Bluetooth-аутентифікатор

USB-аутентифікатор — це апаратний пристрій, який підключається через USB. Його функція полягає у створенні та зберіганні криптографічних ключів у спеціальному (ізольованому) середовищі. При цьому приватний ключ ніколи не покидає пристрою та не може бути скопійований чи зчитаний. Компрометація такого пристрою майже не можлива навіть за умови фізичного доступу. Під час реєстрації автентифікатор генерує криптопару та повертає серверу публічний ключ. В момент входу він підписує челендж і формує криптографічний доказ підтверджуючий особу користувача. Ще однією властивістю USB-аутентификатора є вимога фізичної присутності користувача. Зазвичай реалізується через натискання або сенсорний дотик. Щоб переконатися, що дію виконує людина, а не зловмисне

програмне забезпечення. Тому це робить USB-аутентифікатори стійкими до фішингу та віддалених спроб компрометації.

Криптографія — це набір методів, що дозволяють захищати інформацію шляхом шифрування, роблячи її недоступною для сторонніх. Ідея полягає в тому, щоб дані залишалися конфіденційними та цілісними навіть у сумнівних середовищах. У випадках коли неможливо повністю довіряти сторонам або каналу зв’язку.

Криптографія важливий елемент більшості технологій безпеки. В тому числі для безпарольної автентифікації. Криптографія основується на математичних функціях, які працюють з числами. Зазвичай з значеннями з великим об'ємом. Такі функції створюють односторонні операції, які легко виконати, але зазвичай дуже важко повернути назад. Криптографію поділяють на два напрями: симетрична криптографія, асиметрична криптографія. У симетричній криптографії однаковий ключ використовується для шифрування та дешифрування. Вона швидка та застосовується там, де важлива продуктивність. Наприклад у шифруванні трафіку TLS на етапі обміну великими обсягами даних. Асиметрична криптографія, де використовується криптопара (key pair), яка використовує два різних ключі. Вона дає змогу встановлювати безпечні з’єднання між сторонами, які ніколи не обмінювалися зашифрованою інформацією. Принцип асиметричної криптографії став основою для електронних підписів та механізмів аутентифікації.

Сучасна криптографія також включає певні протоколи. Вони описують, як правильно комбінувати алгоритми, ключі та сертифікати. Протокол WebAuthn об’єднує криптографічні ключі, підтвердження володіння приватним ключем, прив’язку до Origin та захист від фішингу. Схожим чином протокол TLS організовує весь процес встановлення захищеного каналу.

Також варто розглянути криптопару (key pair), як важливий

елемент механізмів безпеки в веб-додатках. Цей елемент актуальний і при інтеграції з такими технологіями як WebAuthn. Криптопара складається з двох взаємопов’язаних ключів. приватного (private key) та публічного (public key). Приватний ключ зберігається в таємниці і зберігається лише у власника. Публічний ключ можна безпечно передавати іншим. Ключі пов’язані таким чином, що перший ключ не можна розшифрувати без другого і навпаки. При цьому знання публічного ключа не дає змогу дізнатися щось про приватний.

Це дає змогу реалізувати дві ключові задачі криптографії:

- Конфіденційність. Якщо хтось хоче надіслати вам зашифроване повідомлення, то він використовує ваш публічний ключ. Розшифрувати його може лише той, хто має приватний ключ. Виходить що навіть якщо публічний ключ відомий всім, повідомлення залишиться секретним.
- Автентичність. Ви можете затвердити дані своїм приватним ключем. Тож будь-хто, хто має ваш публічний ключ, може в цьому переконатись. Оскільки на них буде підпис, який затверджує що дійсно від вас і вони не були змінені.

Це дає значну перевагу порівняно з симетричною криптографією, де однаковий ключ має бути розподілений обом сторонам заздалегідь. Це створює проблеми з безпекою й загалом управлінням ключами. У веб-автентифікації криптопари дозволяють реалізувати безпарольну і безпечну перевірку користувача. Браузер або апаратний ключ генерує пару ключів, де публічний ключ передається на сервер, а приватний лишається локально. При вході користувачу потрібно доводить володіння приватним ключем, не передаючи нічого через мережу. Важливим моментом є надійна генерація та захист приватного ключа. Якщо приватний ключ буде скомпрометовано або випадково втрачено, то втрата доступу може бути незворотною.

Підписання челенджу (challenge) — це криптографічний механізм, який використовується у сучасних протоколах аутентифікації, де необхідно довести приналежність приватного ключа без передачі самого ключа. Процес полягає в тому, що сервер генерує унікальний випадковий рядок (challenge) та надсилає його клієнту. Клієнт вже потім підписує цей челендж своїм приватним ключем. Після цього сервер отримує цифровий підпис і перевіряє його за допомогою відповідного публічного ключа. Цей підхід дозволяє підтвердити автентичність користувача без передачі секретної інформації мережею. Челендж є одноразовим, тому підпис не може бути повторно використаний зловмисником для здійснення атаки. Якщо хтось перехопить підписаний челендж, він не зможе повторити аутентифікацію. Оскільки наступного разу сервер сформує вже новий челендж, який буде відрізнятись від попереднього.

Підписання челенджу забезпечує криптографічну стійкість протоколу. Він основується на використанні асиметричної криптографії, де приватний ключ недоступний нікому крім користувача. Це дозволяє будувати моделі, в яких сервер може досить точно визначити, що саме власник приватного ключа ініціював дію. Навіть не перевіряючи вміст ключа. На прикладі WebAuthn цю операцію виконує браузер у взаємодії з аутентифікатором, Secure Enclave або іншим захищеним середовищем. Важливою є властивість прив’язки челенджу до конкретного походження. Ця властивість вже вбудована в WebAuthn. Це означає, що аутентифікатор підписує об’єднані дані, що включають домен, додаткову інформацію про контекст тощо.

NFC (Near Field Communication) аутентифікатор — це спеціалізований пристрій для безконтактної взаємодії з різними сумісними пристроями під час процесу аутентифікації. Його завдання безпечно зберігати криптографічні ключі та виконувати підписання даних на пристрої, мінімізуючи ризики компрометації. NFC

аутентифікатори зазвичай реалізуються у вигляді карток або компактних пристроїв, які користувач підносить до датчика на пристрої для встановлення безпечного з’єднання. Під час взаємодії з WebAuthn API браузер ініціює обмін даними з аутентифікатором. Потім він повертає необхідний “publicKeyCredential” або підписаний assertion.

Перевага NFC аутентифікаторів полягає у простоті та зручності. Користувачу не потрібно підключати пристрій або вводити дані. Вони забезпечують високий рівень безпеки завдяки апаратному зберіганню ключів та виконанню криптографічних операцій на самому пристрої. Такі аутентифікатори активно застосовуються у корпоративних системах та банківських додатках.

Bluetooth-аутентифікатор — це апаратний пристрій, який використовує бездротовий протокол Bluetooth для взаємодії з пристрєм у процесі автентифікації через WebAuthn. Як і USB-аутентифікатори, він генерує та зберігає криптографічні ключі у захищеному середовищі. Під час реєстрації Bluetooth-аутентифікатор передає серверу публічний ключ, а під час логіну підписує челендж. Бездротовий характер пристрою дозволяє користувачеві проходити автентифікацію без фізичного підключення, що робить його зручним для пристроїв, які не мають USB-портів. Важливим елементом безпеки є механізм присутності користувача. Зазвичай він реалізується через натискання кнопки або сенсорний дотик.

Bluetooth-аутентифікатори підтримують сучасні стандарти FIDO2 та CTAP2, що забезпечує сумісність з веббраузерами. Також з'являється можливість інтеграції для корпоративних SSO систем.

Підсумовуючи можна сказати, що основні принуипи архітектури WebAuthn є здатність взаємодіяти з різними програмними та апаратними компонентами через стандартизовані інтерфейси. API дає змогу веб-додаткам надсилати виклики для створення або підтвердження

ключів, а браузер передає ці запити до аутентифікатора через протокол CTAP. Це дозволяє системам працювати незалежно від конкретної реалізації апаратного пристрою. Тож аутентифікатор, який підтримує FIDO2, може бути успішно інтегрований у веб-додаток. Для побудови таких систем активно застосовуються фреймворки. Вони містять модулі роботи з “publicKeyCredential”, а також готові інтеграції з REST, GraphQL чи gRPC. Фреймворк фактично автоматизує ключові етапи реалізації WebAuthn, що зменшує кількість помилок ng прискорює розробку.

Комунікація між сервером і клієнтом може відбуватися кількома способами. REST забезпечує просту і передбачувану взаємодію через HTTP. GraphQL дозволяє клієнту більш точно описувати потрібні поля, що зменшує надлишкові дані й робить роботу з аутентифікацією оптимізованою. Обидва підходи покладаються на HTTPS, що гарантує захист переданих даних. Якщо важлива висока швидкість і жорсткий контроль структур даних, то використовують Protocol Buffers. Він забезпечує компактний бінарний формат і дозволяє легко змінювати структури повідомлень при цьому без ризику втратити сумісність. Тому Protobuf часто об’єднується з gRPC. gRPC працює з HTTP/2, дозволяючи швидко передавати аутентифікаційні дані. Архітектура RPC приховує мережеві обміни за простими викликами методів, тому розробникам не потрібно вручну формувати запити. Ця інфраструктура працює в межах стандарту FIDO2, який складається з WebAuthn і CTAP. WebAuthn визначає, як веб-додаток взаємодіє з браузером для створення та перевірки ключів. CTAP регулює спілкування між фізичним аутентифікатором і клієнтським пристроєм. Така співпраця забезпечує гнучку архітектуру, де аутентифікаторами можуть бути вбудовані чи зовнішні пристрої.

Важливим елементом безпеки є поняття походження (Origin). Приватний ключ, створений аутентифікатором, прив’язується саме до нього. Походження працює разом з політикою безпеки SOP, яка відмежовує ресурси різних веб-сайтів. SOP блокує спроби втручання на локальну машину. Якщо сайт потребує взаємодії між доменами, для цього застосовується механізм CORS. Він дозволяє серверу вибрати, які зовнішні домени можуть виконувати запити. Перед передачею чутливих даних браузер виконує preflight-перевірку, щоб переконатися, що сервер справді дозволяє такий доступ. В свою чергу WebAuthn не дозволить створити або використати ключ у середовищі, яке не відповідає оригінальному походженню.

WebAuthn поєднує у собі одразу кілька шарів: криптографію, браузерну безпеку, мережеві протоколи та інфраструктурні компоненти. Така архітектура дозволяє створювати масштабовані та стійкі до атак системи. Веб-додатки отримують можливість безпечної взаємодії з апаратними автентифікаторами, а користувач надійний спосіб входу без паролів.

2.2 Загальна архітектура WebAuthn у веб-додатках

Архітектура WebAuthn складається з чотирьох основних компонентів:

- платформа
- браузер
- аутентифікатор
- сервер

Платформа у контексті WebAuthn — це середовище, яке забезпечує взаємодію між браузером та системними компонентами. Вона об’єднує веб-додаток із фізичними або вбудованими механізмами підтвердження особи. Платформа формує безпечні умови для виконання

криптографічних операцій, контролює доступ до ключів та перевіряє правомірність викликів.

У операційних системах платформа зазвичай реалізована як набір системних API. Це може бути Windows Hello, Trusted Platform Module Windows чи вбудований FIDO2-стек. Кожна з цих систем підходить до зберігання приватних ключів враховуючи різні нюанси. Ще платформа виконує функцію маршрутизації. Коли браузер отримує запит створити чи використати credential, то він звертається до платформи через спеціальні інтерфейси. Платформа аналізує, які аутентифікатори доступні в системі. Після цього вона визначає, який механізм варто задіяти, і передає йому challenge та інші параметри. Така архітектура позбавляє веб-додаток необхідності працювати з апаратними драйверами або протоколами взаємодії.

Безпека платформи забезпечується через поєднання системних дозволів, криптографії та перевірок цілісності. Перед виконанням операцій платформа перевіряє, чи має право той хто авторизується використовувати WebAuthn. В цей момент браузер перевіряє чи сторінка відповідає своєму походженню. Це захищає користувача від спроб підробки викликів або перенаправлення операцій на інші сайти. Платформа не дозволить створити чи застосувати credential для домену, який не відповідає походженню ключа. Навіть якщо в системі хтось намагається імітувати запит. Крім цього платформа підтримує життєвий цикл credential. Вона може визначати, чи пристрій здатний обробляти необхідні операції для безпарольної аутентифікації. Тому WebAuthn працює однаково в різних середовищах. При цьому використовуючи специфічні можливості конкретної системи. Тож платформа є центральним компонентом архітектури WebAuthn. Вона забезпечує ізоляцію та адаптацію до різних пристроїв.

Браузер у архітектурі WebAuthn виступає елементом, який поєднує веб-додаток із платформою та аутентифікатором. Він контролює доступ доступу тп інтерпретує стандарти. Забезпечує виконання операції в межах дозволеного контексту. Браузер визначає, чи має конкретна сторінка право ініціювати створення криптографічної пари ключів або запросити підписання challenge. Він перевіряє, щоб запит походив від затвердженого походженням ресурсу. Через браузер реалізується WebAuthn API у вигляді інтерфейсу, через який веб-додатки викликають операції. API працює поверх системного Credential Management, який обмежуючи взаємодію до високорівневих операцій. Коли сторінка викликає “navigator.credentials.create” або “navigator.credentials.get”, браузер формує запит із необхідними параметрами. Після чого виконує Origin-validation і перетворює дані у внутрішній формат, щоб передати їх платформі. В цей період він контролює процес до повернення результату назад у код JavaScript. Важливою частиною його задачі є ізоляція контекстів. У браузері кожен сайт працює в окремому середовищі, і WebAuthn суворо прив’язаний до цього середовища. Ключі, створені для одного домену, недоступні для іншого. Тож сторінка в iframe не може використовувати WebAuthn без спеціальних прав. В тому числі скрипти третіх сторін не мають можливості ініціювати автентифікацію, якщо їх походження відрізняється. Сам браузер контролює цю прив’язку, запобігаючи фішингу навіть у випадках, коли користувач сам перейшов на сайт зловмисників. Також браузер відповідає за взаємодію UI під час операцій з WebAuthn. Він відображає системні діалоги, підказки для біометрії чи підключення автентифікатора. Сайт з малою вірогідністю зможе підробити ці вікна. Браузер виступає активним учасником захисту і посередником між API та платформою.

Браузер адаптує роботу WebAuthn до конкретної платформи. Хоча стандарт єдиний, реалізація в Chrome, Firefox, Safari (та інші)

відрізняється в певних моментах. Не дивлячись на це WebAuthn може працювати на різних пристроях однаково для користувача і одночасно підлаштовуватись під операційну системи. Браузер визначає, коли допустимо використовувати вбудовані сенсори. Він же визначає коли операція має бути заблокована через невідповідність політикам безпеки. Браузер відповідає за стандартизоване виконання процесів у WebAuthn на практиці. Він контролює доступ до API та відповідає за відображення довіреного UI. Завдяки браузеру WebAuthn доступний розробникам без необхідності працювати з низькорівневими механізмами пристрою. Користувач же отримує передбачувану і захищену взаємодію.

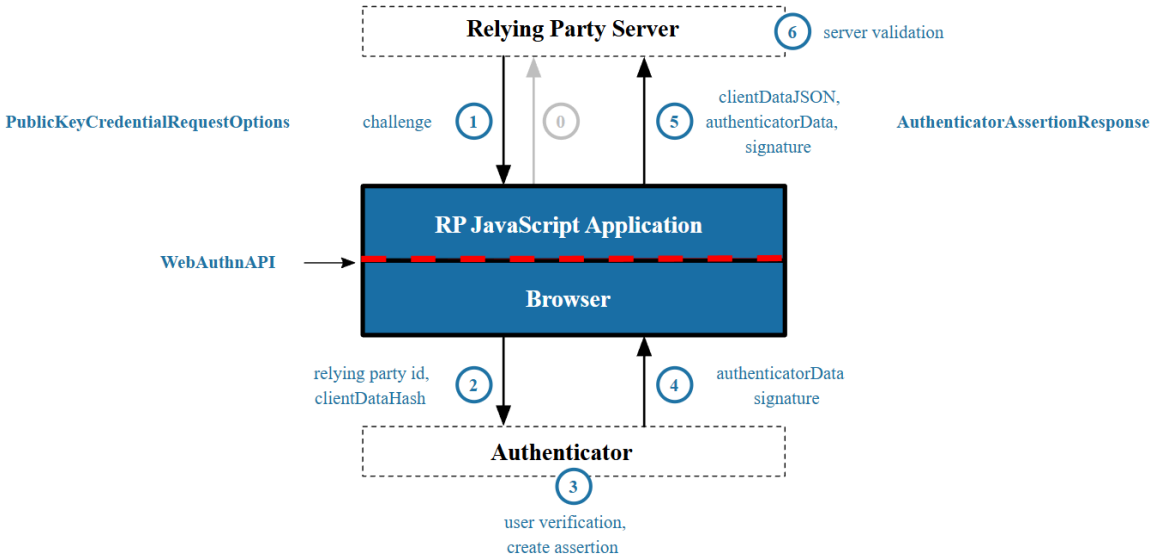


Рис. 2.2 – Процес аутентифікації у WebAuthn

Аутентифікатор у архітектурі WebAuthn є елементом, який відповідає за створення криптографічних ключів та виконання криптографічних операцій в момент аутентифікації. Браузера чи платформа виконують роль посередників, тоді як автентифікатор реалізує базову логіку безпеки. Він генерує пару ключів та підписує challenge під час входу. Також він не дає приватному ключу потрапити у інше середовище системи.

Аутентифікатори поділяються на інтегровані та зовнішні. Інтегровані (platform authenticators), як Windows Hello, Touch ID, Android Biometrics або інші системні механізми ідентифікації. Вони працюють без підключення додаткових пристроїв. Оскільки використовують вбудовані сенсори та отримують доступ до безпечних областей зберігання ключів (Secure Enclave, TPM тощо). Зовнішні або аутентифікатори представляють з себе апаратні ключі на кшталт YubiKey чи Google Titan і під'єднуються через USB, NFC або Bluetooth. Це дозволяє переносити облікові дані між різними пристроями. Обидва типи аутентифікаторів функціонують за однаковими принципами. Крм цього вони підпорядковуються стандартам FIDO2 і CTAP.

У процесі реєстрації автентифікатор створює криптопару. До кожної пари ключів аутентифікатор додає ідентифікатор ключа (credential ID), який дозволяє знайти приватний ключ у сховищі. Тому серверу не потрібно зберігати приватні ключі або спеціальні токени, а лише їх ідентифікатори. Під час входу аутентифікатор отримує challenge від сервера у структурованому вигляді. Він перевіряє, чи дозволена операція для поточного походження і чи відповідає ідентифікатор ключа збереженому ключу. Якщо все коректно, то аутентифікатор виконує підпис, а браузер повертає результат серверу.

Аутентифікатор також відповідає за перевірку користувача через методи "user verification". Це може бути PIN чи інший фактор, залежно від типу пристрою. Аутентифікатор виконує всі ці перевірки локально, не передаючи біометричні дані назовні. Сервер отримує лише факт успішної перевірки. Безпекові властивості автентифікатора визначають, наскільки стійкою є загальна система. Апаратні ключі пропонують високий захист завдяки ізоляції приватного ключа та стійкості до витоку через програмні вразливості, з можливістю працювати у недовіреному середовищі. Інтегровані аутентифікатори забезпечують високий рівень

безпеки завдяки тісній інтеграції з операційною системою та біометричними методами верифікації. Аутентифікатор визначає, наскільки система буде захищена від різних типів кібер атак. Аутентифікатор в архітектурі WebAuthn виконує криптографічні операцій і забезпечує фізичну або програмну захищеність приватних ключів.

Сервер в архітектурі WebAuthn відповідає за логіку ідентифікації користувача та перевірку криптографічних доказів автентичності. Його завдання полягає в обробці запитів та дії з елементом challenge. Сервер зберігає публічний ключ, інформацію про тип аутентифікатора та параметри безпеки. Це виконується як частина моделі “public key credentials”, бо серверу достатньо публічного ключа для подальших перевірок. Якщо користувач робить спробу аутентифікуватись, то сервер генерує виклик, який запобігає повторному використанню даних. Аутентифікатор підписує цей виклик приватним ключем, а сервер перевіряє підпис за допомогою публічного ключа. Сервер контролює політику безпеки, механізми захисту від повторних атак і взаємодію з системами керування сесіями. У системах зазвичай це інтегрується з власними службами IAM або зовнішніми провайдерами, які підтримують WebAuthn. Таким чином сервер виступає останньою ланкою перевірки. Сучасні сервери WebAuthn здатні адаптуватися до різноманітних клієнтських платформ. Вони можуть підлаштовувати вимоги до безпеки залежно від ризиків. Сервер виходить центром керування моделі WebAuthn, забезпечуючи коректну взаємодію між іншими компонентами архітектури.

Архітектура WebAuthn основується на розділенні обов’язків між учасниками. Це створює передбачувану модель взаємодії. Така ізоляція зменшує площу атаки та посилює забезпечує надійність усієї криптографічної схеми. Завдяки такому поділу відповідальності

WebAuthn уникає типових проблем паролів. Коли один скомпрометований елемент може поставити під загрозу всю систему. Кожний компонент виконує лише свій набір задач, які доповнюють один одного. З чого формується децентралізована модель аутентифікації.

Таке розділення обов’язків забезпечує масштабованість та можливість використовувати WebAuthn у різноманітних середовищах. Розробник може посилювати політики безпеки не змінюючи логіку браузера чи платформи.

2.3 Механізми створення криптографічних пар ключів і перевірки підписів

Транспорт ключів у WebAuthn і криптографії означає процес безпечної передачі криптографічних ключів між різними компонентами системи. У WebAuthn транспорт ключа застосовується переважно для забезпечення взаємодії аутентифікатора з платформою та сервером. Щоб прибрати необхідность передавати приватні ключі, які залишаються виключно на пристрої користувача. Принцип базується на асиметричній криптографії. У деяких реалізаціях використовується сертифікат атестації, який доводить походження публічного ключа та його відповідність стандартам безпеки виробника аутентифікатора.

Для безпечного транспортування даних і ключів WebAuthn покладається на перевірені протоколи, такі як TLS. Вони забезпечують шифрування, аутентифікацію сервера та захист від атак MITM. Протокол підтримує відновлення ключів та їх реєстрацію без повторного витоку конфіденційної інформації, що робить систему стійкою навіть при зміні пристрою користувача. У практичному сенсі транспорт ключів у WebAuthn забезпечує сумісністю з різними платформами та зручністю для кінцевого користувача.

Засвідчення (Attestation) — це механізм, що дозволяє серверу переконатися в походженні та надійності аутентифікатора. Він створює

нові ключі для користувача. Під час реєстрації аутентифікатор формує пару ключів, а разом із публічним ключем надсилає Attestation дані. Це структурований пакет, який містить інформацію про виробника, тип аутентифікатора, криптографічний підпис тощо. Цей підпис створюється кореневим атестаційним ключем, який прив'язаний до сертифікаційного центру виробника. Що дозволяє серверу перевірити, чи перед ним не модифікований пристрій.

Механізму полягає в тому, що система може встановити рівень довіри до апаратної чи програмної реалізації, яка генерує криптографічні ключі. Сервер зможе може приймати чи відхиляти реєстрацію на основі атестаційних метаданих, якщо йому важливо працювати лише з сертифікованими апаратними аутентифікаторами. Це дає змогу контролювати безпековий профіль користувачів і застосовувати різні політики з урахуванням рівня захисту.

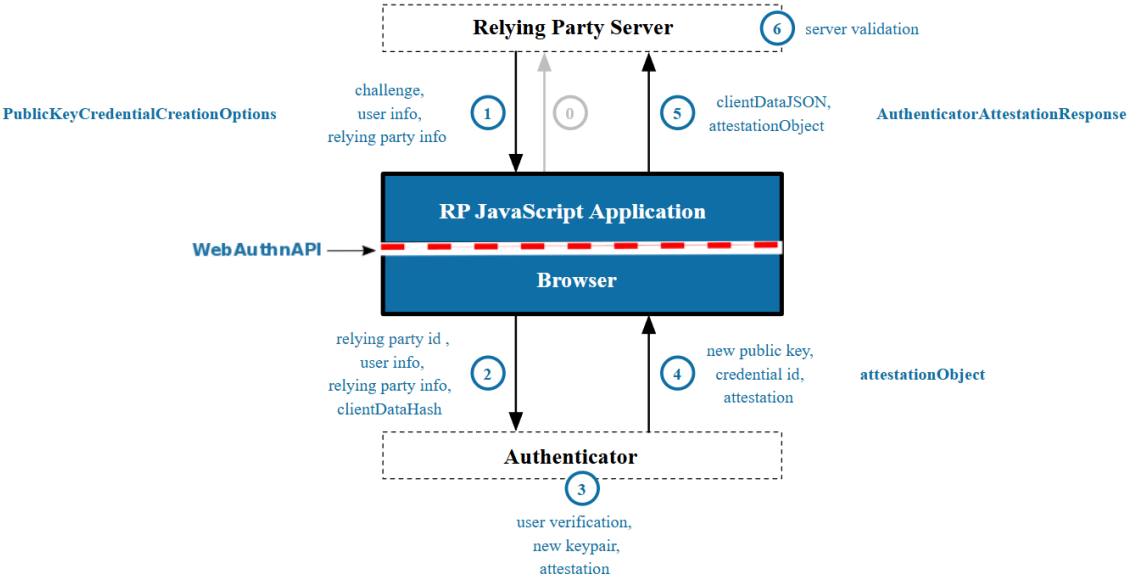


Рис. 2.3 – Процес реєстрації у WebAuthn

В більшості систем Attestation використовується в приватному режимі, коли сервер не прив'язується до моделі аутентифікатора. Він лише перевіряє правильність структури та підпису. Це дає змогу

зберегти приватність користувачів і уникнути ситуацій, коли тип пристрою може бути відстежений за унікальними ключами. Attestation поєднує у собі можливість перевіряти довіру до пристрою та мінімізувати витік персональних даних.

Assertion — це процес підтвердження особи користувача під час входу в систему. Аутентифікатор доводить серверу, що він володіє приватним ключем, створеним під час реєстрації. Цей механізм використовує криптографічний підпис, який аутентифікатор генерує у відповідь на унікальний челендж сервера. Такий челендж щоразу інший, тому повторити або підробити його неможливо. Структура підписаних даних прив’язана до конкретного походження, що заважає атакам повторного використання даних. Отримавши відповідь аутентифікатора, сервер виконує два ключові кроки. Він перевіряє підпис з використанням публічного ключа та аналізує додаткові дані, включаючи лічильник використань (signCount), інформацію про автентичність клієнтських даних і прив’язку до домену. Якщо все відповідає очікуваним параметрам, то сервер вважає користувача успішно аутентифікованим. Це створює захищений канал без передачі приватних ключів і без залежності від паролів.

Також Assertion враховує контекст користувача. Поєднання локальної верифікації та криптографічного підпису робить цю процедуру не тільки безпечною, а й швидкою. Аутентифікація відбувається швидко та без необхідності вводити пароль або підтверджувати дані. Тож Assertion є важливим інструментом у WebAuthn, що перетворює криптографічну модель аутентифікації на простий і зручний процес для кінцевого користувача.

Алгоритми цифрового підпису використовуються в системах аутентифікації. Вони визначають спосіб створення надійного

криптографічного доказу володіння приватним ключем. Найпоширеними є три підходи: RSASSA-PKCS1-v1_5, ECDSA та EdDSA. Кожен із них представляє окрему криптографічну традицію й відрізняється в різних аспектах.

RSASSA-PKCS1-v1_5 основуєтьяс на RSA-криптографії та вважається одним із найширше підтримуваних алгоритмів. Він простий у реалізації, забезпечує високу сумісність і залишається актуальним у корпоративних інфраструктурах з RSA. Його ключі більші за ключі процедур на еліптичних кривих, а операції займають більше часу. Це робить RSA менш ефективним у мобільних або вбудованих пристроях. Незважаючи на надійність, цей алгоритм поступово витісняється більш просунутими схемами через нижчу продуктивність і досить громіздку криптографічну структуру.

ECDSA використовує математику еліптичних кривих, що дозволяє отримати криптографічну стійкість при набагато коротших ключах і вищій швидкодії. Тому ECDSA став основою багатьох сучасних стандартів, включно з FIDO2 та WebAuthn. Він оптимально поєднує безпеку та продуктивність. Також зменшує навантаження на апаратні ресурси та покращує час обробки підписів. ECDSA чутливий до якості генератора випадкових чисел. Якщо випадкові значення під час підпису будуть передбачуваними, то можливе відновлення приватного ключа. Тому реалізації повинні бути особливо захищеними і гарно сформованими.

EdDSA — це відносно новий алгоритмів і представлений такими кривими, як Ed25519. Він спроектований з урахуванням багатьох недоліків попередніх схем. Він забезпечує стабільну швидкість, має стійкість до збоїв при генерації випадковостей і демонструє високий

рівень безпеки навіть на малопотужних пристроях. EdDSA використовує детермінований підхід до підпису. Результат він формує на основі приватного ключа та повідомлення, без залежності від зовнішнього джерела випадкових чисел. Це знижує ризики пов’язані з неправильним генеруванням ентропії. Тому робить алгоритм більш передбачуваним та безпечним у практичних застосуваннях.

Тож RSASSA-PKCS1-v1_5 залишається актуальним завдяки своїй сумісності, але його використовують дедалі рідше через незручність. ECDSA пропонує високу ефективність та оптимальне співвідношення швидкості й безпеки, але потребує обережності в реалізації. EdDSA ймовірно найбільш перспективний серед них. Він легкий, швидкий та менш схильний до реалізаційних помилок. Тому добре підходить для систем, орієнтованих на сучасну безпарольну автентифікацію.

2.4 Основні типи аутентифікаторів у WebAuthn

Платформенні аутентифікатори — вони вбудовані в пристрій для двофакторної або безпарольної аутентифікации. Вони відрізняються тим, що не потребують підключення зовнішніх пристроїв і працюють тісно з операційною системою та браузером. Такий спосіб дозволяє користувачам проходити аутентифікацію легко та швидко. Приватні ключі, необхідні для підписів, зберігаються у захищеному середовищі пристрою. Це гарантує, що ключ ніколи не покидає пристрій і не стає доступним для шкідливого програмного забезпечення. Platform-аутентифікатори зазвичай підтримують локальну та асиметричну аутентифікацію користувача. Наприклад через PIN, відбиток пальця тощо або асиметричну криптографію для взаємодії із серверами. Це дозволяє реалізувати високий рівень безпеки без необхідності додаткових апаратних токенів. Такий тип аутентифікатора може підійти для мобільних додатків, веб-сервісів та корпоративних

систем. Перевагою Платформенних аутентифікаторів є поєднання надійного захисту криптографічних ключів із максимальною інтеграцією у вже наявну платформу.

Міжплатформні аутентифікатори (Cross-platform) — це зовнішні пристрої, які дозволяють користувачам безпечно проходити аутентифікацію. Вони можуть підключатися до різних пристроїв через USB, NFC, Bluetooth. Тож це робить їх незалежними від апаратного середовища. Їхня функція полягає у генерації та зберігання приватних ключів у захищеному середовищі пристрою. Вони підтримують різні методи підтвердження особи користувача. Наприклад натискання кнопки на токені, сканування відбитка пальця або введення PIN-коду. Вони особливо корисні для користувачів, які працюють на кількох пристроях або в корпоративних середовищах із різноманітними операційними системами. Оскільки один аутентифікатор можна використовувати на різних пристроях без додаткової інтеграції. Міжплатформні аутентифікатори дозволяють користувачам залишатися мобільними та автономними.

Роумінг (Roaming) аутентифікатори — це тип зовнішніх аутентифікаторів, які дозволяють користувачам проходити безпечну автентифікацію на будь-якому сумісному пристрої. Вони зазвичай підключаються через USB, NFC або Bluetooth. Що дозволяє їм бути універсальними і використовувати один токен на пристроях різних виробників. Для підтвердження особи користувача цей токен може все те саме, що і міжплатформені аутентифікатори. Roaming аутентифікатори корисні у корпоративних або мобільних середовищах, де користувачі часто змінюють пристрої. Вони дозволяють легко переносити аутентифікаційні дані без потреби у повторній реєстрації.

Вбудовані (Built-in) аутентифікатори — це вбудовані в пристрій апаратні або програмні модулі, які забезпечують безпечну

автентифікацію. Вони роблять можливою аутентифікацію без необхідності використання зовнішніх токенів. Вони можуть бути інтегровані у ноутбуки, смартфони, планшети. Зазвичай у вигляді сканера відбитків пальців, розпізнавання обличчя або спеціального апаратного чіпа, що відповідає за зберігання криптографічних ключів. Вони практичні для персональних пристроїв, де важлива швидкість і простота входу. Вони підтримують стандартні механізми WebAuthn, забезпечуючи сумісність з веб-сервісами та застосунками.

2.5 Механізми забезпечення безпечної аунтефікації з WebAuthn

Одна з переваг WebAuthn є захист від фішингу, що забезпечує захист від спроб шахрайського отримання облікових даних користувача. Звичайні паролі легко піддаються націленим атакам або коли користувач вводить їх на шахрайському сайті створеному для крадіжки інформації. WebAuthn вирішує цю проблему через криптографічні ключі, які ніколи не покидають аутентифікатор і не передаються веб-сайтам у відкритому вигляді. Аутентифікатор підписує запит лише для конкретного походження, що робить неможливим використання ключів на шахрайських сайтах. Якщо користувач все ж перейде на фішинговий ресурс, то приватний ключ не буде задіяний, а спроба входу буде відхилена. Такий підхід підвищує безпеку та довіру користувачів. Він виключає людський фактор, який часто стає слабким місцем при використанні паролів.

Також за протидію атакам у WebAuthn відповідає Origin-binding, який гарантує, що аутентифікаційні ключі прив’язані до конкретного веб-ресурсу. Origin визначається як поєднання протоколу (наприклад, HTTPS), доменного імені та порту. Це призводить до того що навіть якщо зловмисник спробує змусити користувача використати свій ключ фішинговому або підробленому сайті, то це ні до чого не призведе. Аутентифікатор просто не підпише запит, для якого ключ був створений,

бо Origin не співпадає. Завдяки origin-binding забезпечується достатній захист від атак типу Man-in-the-Middle та надійний захист від фішингу. Приватний ключ користувача ніколи не залишає аутентифікатор і не передається веб-сайтам у відкритому вигляді.

Атаки типу MITM (Man-in-the-Middle) виникають тоді, коли третя сторона отримує змогу перехоплювати, аналізувати або змінювати трафік між користувачем та сервером. MITM може відбуватися через підроблені точки Wi-Fi, фальшиві DNS-записи чи скомпрометовані маршрутизатори. Іноді через невірно встановлені сертифікати. У таких умовах зловмисник може отримувати паролі, перехоплювати токени доступу. Також змінювати вміст сторінок або перенаправляти користувача на фішингові ресурси.

TLS виступає базовим захисним шаром у взаємодії браузера з веб-сервером, забезпечуючи цілісність, шифрування та аутентичність трафіку. Він створює захищений канал, у якому дані не можуть бути змінені третьою стороною. Під час встановлення TLS з’єднання клієнт перевіряє дійсність сертифіката сервера . Коли переконується у справжності домену та домовляється про криптографічні параметри сесії. Такий підхід дозволяє уникнути ситуацій, коли зловмисник підміняє сервер або втручається в комунікацію. TLS значно знижує ризик MITM атаки, адже навіть якщо зловмисник спробує підмінити трафік, він не зможе пройти перевірку сертифіката. Хоча загрози все одно існують коли користувач ігнорує попередження браузера або коли атака здійснюється через скомпрометований кореневий сертифікат. Тому сучасні протоколи аутентифікації використовують TLS у поєднанні з додатковими механізмами. Разом ці технології створюють багатошаровий захист.

2.6 Сумісність з браузерами, операційними системами та апаратними ключами

У WebAuthn широка сумісність з браузерами, операційними системами та апаратними автентифікаторами. Це дає можливість технології бути придатною для масового використання у веб-додатках. Забезпечення сумісності дозволяє реалізувати безпарольну автентифікацію без необхідності встановлення додаткового програмного забезпечення та значного втручання в інфраструктуру.

Сумісність WebAuthn із браузерами є важливим для широкого впровадження стандарту аутентифікації без паролів. Більшість сучасних браузерів підтримують WebAuthn, такі як: Google Chrome, Mozilla Firefox, Microsoft Edge та Apple Safari. Користувачі можуть проходити аутентифікацію через апаратні або вбудовані аутентифікатори незалежно від обраного браузера. Підтримка WebAuthn у браузерах включає роботу з різними типами ключів, керування запитами на аутентифікацію та реєстрацію тощо. Базові функції стандарта реалізовані майже всюди, деякі специфічні можливості можуть відрізнятися залежно від браузера та платформи. Тому при впровадженні WebAuthn варто перевіряти сумісність не тільки з самим браузером, а й із його версією. Також перепровіряти підтримку апаратних пристроїв, щоб гарантувати стабільну роботу для всіх користувачів.

Сумісність WebAuthn з операційними системами теж важлива для забезпечення універсальної та безпечної аутентифікації. Операційні чичтеми (ОС), такі як Windows, macOS, Linux підтримують WebAuthn у поєднанні з відповідними браузерами та апаратними аутентифікаторами. Системи Android і iOS теж мають пітримку технології.

- На Windows стандарт інтегрований через Windows Hello, що дозволяє користувачам використовувати біометрію або PIN-код для автентифікації.
- У macOS і iOS користувачі можуть застосовувати Touch ID або Face ID.

- Android забезпечує підтримку через системні API для біометрії та зовнішніх пристроїв, а також через сумісні браузери.
- Linux також підтримує WebAuthn через сучасні браузери та зовнішні апаратні ключі.

Рівень інтеграції та підтримка різних типів аутентифікаторів може відрізнятися залежно від версії ОС та конкретного пристрою. Тому при впровадженні WebAuthn у корпоративних або масових сервісах потрібно враховувати різноманітність ОС користувачів та тестувати роботу аутентифікації на різних платформах.

Сумісність WebAuthn із апаратними ключами є не менш важливим фактором для забезпечення безпечної аутентифікації без паролів. Апаратні ключі, такі як YubiKey, Feitian або інші сертифіковані FIDO2-пристрої. Вони можуть підключатися до комп'ютера або мобільного пристрою через NFC, USB, або Bluetooth. WebAuthn підтримує більшість сучасних апаратних ключів, проте для повної сумісності важливо враховувати тип інтерфейсу та специфікації FIDO2, які реалізовані виробником. Під час інтеграції слід перевіряти, чи пристрій правильно працює з браузерами та операційними системами користувачів. Також чи він підтримує реєстрацію і аутентифікацію. Це робить апаратні ключі досить універсальним і безпечним рішенням для корпоративних та особистих облікових записів.

Висновок до розділу 2

Стандарт WebAuthn формує інфраструктуру де платформа та аутентифікатор працюють як єдина система. Його архітектура поєднує набір API, криптографічні механізми та правила безпеки браузерного середовища, створюючи модель взаємодії, що мінімізує людський фактор. Стандарт працює у зв'язці з FIDO2, що поєднує WebAuthn і CTAP дозволяючи використовувати різні апаратні автентифікатори. У

WebAuthn браузер керує взаємодією між веб-додатком і аутентифікатором. Він використовує визначений набір API, через який виконує необхідні операції. При цьому дані передаються лише в межах дозволеного походження. API знімає з розробника необхідність прямої роботи з апаратними пристроями. Тому WebAuthn інтегрується в застосунки легше, ніж класичні парольні схеми з OTP або SMS. На клієнті система працює через WebAuthn API у браузері та CTAP для комунікації з автентифікаторами. В той час як на сервері використовуються сервіси основані на REST, GraphQL або RPC для прийому даних і управління обліковими записами.

Для реалізації аутентифікації у бекенд-системах застосовуються фреймворки, які надають готові модулі для створення валідації підписів, менеджменту сесій чи токенів. Вони використовують HTTP(S) та структури REST, де транспортом даних виступають JSON повідомлення. Такий підхід дозволяє зберегти простоту впровадження й одночасно підтримати масштабованість. GraphQL додає можливість точково контролювати, які поля потрібні клієнту під час реєстрації або входу. У середовищах з високим навантаженням, де обсяг переданих даних надто високий, використовуються Protocol Buffers. Він дозволяє швидше (ніж JSON) обробляти структури і зберігає типізованість. Protobuf інтегрується з фреймворком gRPC, що базується на HTTP/2 та RPC-викликах. RPC ховає всі мережеві деталі за викликами функцій. Такий підхід дає можливість ефективно розподіляти логіку між мікросервісами й будувати системи, які обробляють повідомлення WebAuthn швидко та без втрати цілісності. Застосування gRPC у WebAuthn зменшує навантаження на сервери за рахунок бінарних форматів і додаткових можливостей HTTP/2.

У системі FIDO2 важливим елементом виступає концепція походження (Origin). Воно складається зі схеми, домену та порту. Також

визначає контекст, у якому дозволені операції. Приватний ключ, створений аутентифікатором, завжди прив’язаний до конкретного походження. Ця властивість майже повністю усуває ризики фішингу.

До системи контролю доступу належать механізми SOP і CORS. SOP забезпечує ізоляцію сайтів між собою, не дозволяючи скриптам одного походження читати або модифікувати ресурси іншого. Це запобігає викраденням токенів, крадіжці сесій та маніпуляціям через iframe. CORS дозволяє точно визначити, яким походженням можна виконувати запити, та які типи даних допускаються для передачі. Таким чином формується керована модель , де сервер встановлює правила, а браузер суворо їх виконує.

Об’єднання всіх цих механізмів створює структуру, у якій WebAuthn працює як цілісна система. Ланцюг взаємодії побудовано так, щоб відмовостійкість була закладена на кожному рівні. У сучасній інфраструктурі WebAuthn стає стандартом для організацій, які прагнуть відмовитися від паролів і перейти до моделей на основі криптографічних ключів. Механізми WebAuthn утворюють технічну основу, яка дозволяє впроваджувати сильну, стійку до атак аутентифікацію без ускладнення досвіду користувача.

3 ІНТЕГРАЦІЯ ТЕХНОЛОГІЇ WEBAUTHN У ВЕБ-ДОДАТОК

3.1 Архітектурні принципи інтеграції WebAuthn у веб-додаток

Інтеграція WebAuthn у веб-додаток варто починати з проєктування архітектури, яка має врахувати взаємодію між клієнтом, сервером та автентифікаторами. Архітектура WebAuthn працює не як типова модель введення, у ній роль браузера, системної платформи й сервера чітко окреслена. Тому якісний дизайн на початковому етапі визначає надійність реалізації. У типових веб-додатках сервер відповідає за перевірку пароля та зберігає хеші в базі даних. У WebAuthn логіка змінюється. Ключі зберігаються виключно у користувача, а сервер управляє лише публічними ключами та метаданими credential-обліковок. Це змушує переглянути архітектуру модуля аутентифікації. Необхідно передбачити окремий компонент, який формує "challenge", валідує підпис, перевіряє параметри походження, тип автентифікатора, рівень довіри до пристрою та відповідність формату WebAuthn. Тому сервер стає грамотним операцій, які здійснює користувач через браузер. WebAuthn API, вбудоване безпосередньо в браузер, виконує функції посередника між веб-додатком і аутентифікатором. Клієнт концептуально стає частиною криптографічної моделі. Від браузера залежить, який автентифікатор буде вибраний і чи коректно формується структура credential. Тому концептуальна архітектура клієнтського рівня має включати аналіз середовища користувача, особливості різних браузерів і обмеження платформ: Windows Hello, Touch ID, Android Biometrics або зовнішні ключі FIDO2.

Комунікація між клієнтом і сервером повинна бути спроєктована як передбачуваний цикл взаємодій. Реєстрація та логін WebAuthn завжди працюють через двофазну модель. Спочатку сервер генерує параметри, потім клієнт викликає автентифікатор, далі браузер формує credential та транспортує його назад. API має бути чітко визначеним, а структура

переданих даних стабільною. Архітектура повинна враховувати обробку помилок на різних рівнях. Від відсутності автентифікатора до невідповідності challenge чи спроби використати credential на іншому домені. Ще один важливий аспект це інтеграція WebAuthn із загальною системою управління користувачами. Потрібно заздалегідь визначити, як користувач зберігатиме кілька credential-ключів, як виконуватиметься прив’язка пристрою до акаунта, як реалізувати відновлення доступу у випадку втрати ключа. Ці рішення впливають UX-модель, тому мають бути закладені на рівні проектування. Архітектура також повинна враховувати профіль загроз. WebAuthn значно зменшує ризики атак, але все ще залежить від загальної безпеки серверної інфраструктури. Потрібно закладати механізми захисту від MITM на рівні TLS до контролю доступу над API. Крім цього перевірки коректності параметрів, що надходять з клієнта.

Архітектурні принципи інтеграції WebAuthn представляють з себе комплексне проектування системи, де з самого початку потрібно враховувати користувацької взаємодії та вимоги до надійної серверної частини. Це дозволяє створити інфраструктуру, яка забезпечує стійкість до атак та простоту для користувача, з можливістю масштабування в майбутньому.

3.2 Серверна сторона веб-додатку з WebAuthn

Серверна частина інтеграції WebAuthn відповідає за генерацію викликів (challenge), перевірку криптографічних даних, зв’язок з базою даних та забезпечення довготривалої консистентності credential-записів.

Побудова серверного WebAuthn-шару зазвичай починається з виокремлення спеціалізованого сервісу, який відповідає за роботу з credential-обліковими даними. Він генерує challenge для реєстрації та логіна, забезпечує підписання правильних параметрів та перевіряє отриманий від клієнта підпис. Цей сервіс працює разом з

API-контролерами та гарантують відповідність специфікації FIDO2. Варто ізолювати структуру від API, за для стабільної доменної логіки.

У процесі обробки credential-даних сервер виконує певний перелік криптографічних перевірок. Він звіряє challenge, перевіряє правильність походження підпису, а також аналізує структуру authenticatorData та clientDataJSON. Щоб потім затвердити, що операцію виконано саме тим аутентифікатором, який був зареєстрований користувачем. На цьому етапі важливо правильно працювати з COSE-ключами, форматами публічних ключів та атрибутами безпеки, включно з прапорами (userPresent, userVerified) і значеннями для визначення можливих спроб відтворення (replay).

Сервер зберігає тільки публічні ключі, пов’язані атрибути (credentialId, тип ключа, алгоритм, counter, інформацію про платформний чи крос-платформний автентифікатор), а також метадані для подальшої перевірки. База даних має бути оптимізована для швидкого пошуку credential. Архітектурно поширеною практикою є використання окремої таблиці або колекції для WebAuthn credentialів, з чітким розділенням.

Впровадження додаткових механізмів безпеки на сервері є обов’язковим. Сюди відносяться rate limiting для операцій створення та підтвердження credential-даних. Не менш важливе використання HTTPS, коректна робота з origin та RP ID, а також перевірка метаданих автентифікаторів через FIDO Metadata Service. Це дозволяє виключити ненадійні або скомпрометовані пристрої.

З архітектурної точки зору сервер повинен бути готовим до горизонтального масштабування: зберігати challenge в Redis або іншому швидкому key-value сховищі. Також забезпечувати ідентичну поведінку між інстансами та синхронізацію counter-даних. Це має вирішальне значення для високонавантажених систем, де WebAuthn використовується як механізм входу.

3.3 Клієнтська частина веб-додатку з WebAuthn

Клієнтська частина інтеграції WebAuthn визначає, як веб-додаток взаємодіє з аутентифікатором через JavaScript API. Тут реалізується логіка ініціації операцій створення нового аутентифікатора або підтвердження особи користувача. Робота з WebAuthn у браузері зводиться до коректного використання механізмів navigator.credentials та узгодження параметрів сервера.

Концептуально клієнтська частина отримує від сервера необхідні challenge-дані, формує запити у форматі PublicKeyCredentialCreationOptions або PublicKeyCredentialRequestOptions. Також передає їх до navigator.credentials.create чи navigator.credentials.get. Цей API забезпечує взаємодію між браузером і аутентифікатором.

Одним із завдань клієнта є коректна обробка бінарних даних, зокрема challenge, user.id, credentialId та COSE-структур. Ці елементи надходять у вигляді Base64URL або ArrayBuffer, і браузерний рівень має забезпечити їх правильну конвертацію. Будь-яка помилка у форматуванні даних призведе до неможливості створення або використання ключів. Тому на цьому етапі архітектура потребує чіткої узгодженості між сервером та клієнтом.

Коли navigator.credentials.create успішно викликається, то автентифікатор генерує пару ключів і повертає публічні дані разом із цифровим підписом. Клієнту потрібно зібрати ці дані та перетворити їх у формат, що відповідає вимогам серверного API. Після чого надіслати назад на backend для перевірки та збереження credential-запису. Аналогічна логіка застосовується під час входу, тільки замість створення ключа виконується операція підписання сертифікованих даних, де аутентифікатор використовує свій приватний ключ.

Крім криптографічної взаємодії, клієнтська частина відповідає за UX та поведінку інтерфейсу. Веб-додаток повинен передбачити можливі стани системи. Налагодження інформативних повідомлень та плавних сценаріїв взаємодії дозволяє WebAuthn працювати як гармонійна частина інтерфейсу.

Суттєвою частиною концептуального проектування є розуміння варіативностей середовища. Наприклад, iOS та Android мають різні специфічні обмеження щодо cross-origin credentialів. Браузери в свою чергу реалізують WebAuthn API з невеликими відмінностями. Це вимагає передбачення fallback-сценаріїв, а також модульної реалізації клієнтського шару. Для подальшого розширення без ризику порушити вже існуючу інтеграцію.

Клієнтська частина WebAuthn формує рівень системи, де користувач безпосередньо взаємодіє з технологією. Вона поєднує криптографію, мережеві запити, роботу з асинхронними операціями та обробку помилок та UX. Саме на цьому рівні технологія набуває глибокого рівня інтеграції, коли користувач не задумується про складність, а просто виконує дію для аутентифікації.

3.4 Процес реєстрації та логіна у веб-дадатку з WebAuthn

Процеси реєстрації та логіна у WebAuthn формують логіку взаємодії користувача з аутентифікаційною системою. На концептуальному рівні ці процеси є двома взаємопов’язаними фазами роботи криптографічної моделі.

```
if (!window.PublicKeyCredential) { /* Client not capable. Handle error. */ }

// credentialId is generated by the authenticator and is an opaque random byte array
var credentialId = new Uint8Array([183, 148, 245 /* more random bytes previously generated
var options = {
  // The challenge is produced by the server; see the Security Considerations
  challenge: new Uint8Array([4,101,15 /* 29 more random bytes generated by the server */]),
  timeout: 120000,  // 2 minutes
  allowCredentials: [{ type: "public-key", id: credentialId }]
};

navigator.credentials.get({ "publicKey": options })
    .then(function (assertion) {
    // Send assertion to server for verification
}).catch(function (err) {
    // No acceptable credential or user refused consent. Handle appropriately.
});
```

Рис. 3.1 – Перший приклад реалізації аутентифікації у WebAuthn

```
if (!window.PublicKeyCredential) { /* Client not capable. Handle error. */ }

var encoder = new TextEncoder();
var acceptableCredential1 = {
    type: "public-key",
    id: encoder.encode("BA44712732CE")
};
var acceptableCredential2 = {
    type: "public-key",
    id: encoder.encode("BG35122345NF")
};

var options = {
  // The challenge is produced by the server; see the Security Considerations
  challenge: new Uint8Array([8,18,33 /* 29 more random bytes generated by the server */]),
  timeout: 120000,  // 2 minutes
  allowCredentials: [acceptableCredential1, acceptableCredential2],
  extensions: { 'credProps': true }
};

navigator.credentials.get({ "publicKey": options })
    .then(function (assertion) {
    // Send assertion to server for verification
}).catch(function (err) {
    // No acceptable credential or user refused consent. Handle appropriately.
});
```

Рис. 3.2 – Другий приклад реалізації аутентифікації у WebAuthn

Реєстрація починається з того, що сервер генерує криптографічний challenge та опис параметрів credential-профілю, який буде створений. Ці

параметри передаються на клієнтську частину, де браузер ініціює процедуру створення нового автентифікатора через WebAuthn API. Користувач підтверджує намір взаємодією з обраним аутентифікатором. В цей момент створює зв’язок між користувачем, пристроєм та веб-додатком. У результаті автентифікатор створює пару ключів. Сервер перевіряє підписану відповідь, валідуючи ключі, автентифікатор та параметри безпеки. Успішна реєстрація завершується збереженням ідентифікатора credential разом з публічним ключем. Цей етап формує початкову довіру між системою та користувачем і створює криптографічний профіль для майбутніх входів. Процес логіна працює за схожим принципом Підтверджуючи що користувач контролює приватний ключ, створений під час реєстрації. Сервер генерує новий challenge, надсилає його на клієнт. Потім браузер ініціює операцію підписання. Аутентифікатор отримує запит, запитує у користувача підтвердження, виконує криптографічний підпис та повертає його клієнту. Сервер верифікує підпис за відповідним публічним ключем, і якщо все коректно то користувач успішно аутентифікований.

Обидва процеси можна розглядати як два етапи однієї моделі довіри. Реєстрація відповідає за створення унікальної, захищеного зв'язку між користувачем і системою. В свою чергу логін відповідає за підтвердження контролю над цим зв'язком. Система від початку позбавлена потреби зберігати таємну інформацію або обробляти паролі, що значно спрощує архітектуру безпеки. Для розробника моделювання цих процесів вимагає продуманої взаємодії між клієнтом і сервером та адаптації під різні типи аутентифікаторів. Від якості реалізації цих процесів залежить успішність усього впровадження WebAuthn у веб-додаток.

3.5 Інтеграція з OAuth 2.0, OIDC, SSO-провайдерами.

Інтеграція WebAuthn із системами OAuth 2.0, OpenID Connect та корпоративними SSO-провайдерами формує більш складну, але сильнішу архітектурну модель автентифікації. Це дозволяє об’єднати локальну криптографічну автентифікацію користувача з глобальними механізмами керування доступом.

У OAuth 2.0 WebAuthn може працювати як механізм аутентифікації, що замінює парольну взаємодію під час видачі токенів доступу. Цей підхід створює модель, в якій аутентифікація виконується локально і криптографічно. Авторизація виконується через стандартизовані токени, що легко інтегруються з багатьма сервісами. OpenID Connect робить інтеграцію більш структурованою, адже надає чітку модель ідентичності, user-info та профільних атрибутів. WebAuthn використовується для підтвердження особи при створенні ID-токену, гарантуючи, що ідентичність прив’язана до реального контролю над приватним ключем. Цей зв’язок усуває класичні ризики злочинних дій. Така інтеграція створює надійну іеформацію про користувача, яку довірені додатки можуть безпечно використовувати. У корпоративних SSO-системах, таких як Azure AD, Okta або Keycloak. WebAuthn набуває властивостей одного із факторів автентифікації або навіть замінником паролів. Це дозволяє організаціям централізовано керувати політиками безпеки, одночасно спрощуючи доступ до різних внутрішніх сервісів. Така комбінація усуває потребу в паролях на рівні всієї компанії знижує операційні ризики.

На рівні інтеграції WebAuthn із цими протоколами перетворює аутентифікацію на модульний компонент загальної цифрової інфраструктури, який може використовуватися всередині іншої архітектурної моделі. В якій присутні токени, сесії чи розподілені сервіси. Така модель гарно масштабується, адаптується для мобільних

додатків та підтримує різні види аутентифікаторів. Завдяки такій інтеграції WebAuthn формує швидку аутентифікацію

Висновок до розділу 3

Архітектурне проектування починається з чіткої організації взаємодії між клієнтом, сервером і автентифікаторами, де браузер і клієнтська платформа стають частиною криптографічної моделі. Також сервер відповідає за управління публічними ключами, challenge-даними та метаданими credential-обліковок. Такий підхід дозволяє забезпечити безпечну обробку даних користувача та стійкість до атак типу MITM, фішинг або replay.

Серверна частина реалізує логіку перевірки credential, управління challenge, обробку ключів COSE та зберігання публічних ключів у структурованій базі даних. Важливим аспектом є забезпечення масштабованості, синхронізації counter-даних і захист від високонавантажених сценаріїв використання. Додаткові механізми безпеки, такі як HTTPS, rate limiting, перевірка RP ID та FIDO Metadata Service, підвищують надійність системи. Клієнтська частина відповідає за взаємодію з аутентифікатором через WebAuthn JavaScript API та “navigator.credentials”, обробку бінарних даних. Клієнт забезпечує злагоджену роботу криптографічних операцій, що покращує UX. Процеси реєстрації та логіна у WebAuthn формують дві взаємопов’язані фази. Ці процеси позбавлені необхідності зберігати паролі на сервері, що спрощує архітектуру безпеки та підвищує стійкість системи.

Інтеграція WebAuthn з протоколами OAuth 2.0, OpenID Connect та провайдерами SSO дозволяє об’єднати локальну аутентифікацію з механізмами управління доступом. Це створює модульну, масштабовану та безпечну модель. WebAuthn виступає як надійний фактор аутентифікації або замінник паролів, забезпечуючи централізоване управління безпекою та зручність користування у корпоративних та

комерційних системах. Планування інтеграції технології WebAuthn у веб-додатки є багаторівневою і вимагає продуманого проектування на рівні архітектури, серверного та клієнтського коду. Якісне проектування архітектури зможе забезпечити високу надійність і зручність користування для кінцевого користувача.

4 ОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ РОЗВИТКУ WEBAUTHN

4.1 Порівняльний аналіз WebAuthn з іншими безпарольними технологіями

Порівняльний аналіз WebAuthn з іншими безпарольними технологіями показує, що кожен підхід має свої сильні та слабкі сторони, і вибір оптимального методу залежить від вимог безпеки. WebAuthn вирізняється високим рівнем захисту від фішингу завдяки механізму прив’язки до конкретного веб-оригіну, що робить його більш надійним у порівнянні з одноразовими паролями (OTP). На відміну від SMS-OTP, WebAuthn не залежить від каналів передачі, які можуть бути перехоплені. У порівнянні з біометричними системами на мобільних пристроях WebAuthn пропонує більш уніфіковану платформу для різних пристроїв і браузерів. Також дає розробникам контроль над процесом реєстрації та автентифікації без необхідності покладатися на пропрієтарні API виробників. Біометрія може бути зручною, але часто не універсальна через різноманіття пристроїв і може викликати проблеми з приватністю.

Інші безпарольні технології, такі як одноразові коди через TOTP або апаратні токени, також забезпечують підвищену безпеку, але потребують додаткових кроків для користувача. WebAuthn поєднує переваги апаратних токенів і біометрії, забезпечуючи одночасно простоту для користувача. Також надійний захист від атак та стандартизовану взаємодію між браузером, платформою та сервером. WebAuthn вигідно відрізняється за критеріями безпеки та гнучкості, оскільки дозволяє уникнути поширених проблем безпарольних систем,зберігаючи зручність використання та підтримку різних типів аутентифікаторів.

4.2 Оцінка юзабіліті, швидкодії, стабільності

Оцінка юзабіліті, швидкодії та стабільності WebAuthn свідчить про те, що ця технологія надає користувачам досить інтуїтивний і зручний досвід аутентифікації. Водночас забезпечуючи високий рівень безпеки. З точки зору юзабіліті, WebAuthn усуває необхідність введення паролів, що спрощує процес доступу до сервісів. Користувачі можуть швидко реєструвати аутентифікатори та входити в систему одним дотиком або за допомогою біометричних даних. У швидкодії WebAuthn демонструє високий рівень відгуку, оскільки процес генерації та перевірки ключів відбувається локально на пристрої користувача. Передача даних на сервер обмежується мінімальним набором необхідної інформації. Це знижує затримки у порівнянні з типовими методами двофакторної аутентифікації, де потрібне підтвердження через сторонні додатки.

Стабільність системи забезпечується стандартизованою архітектурою WebAuthn та підтримкою різних типів аутентифікаторів. Протокол витримує різні сценарії використання, такі як одночасне підключення декількох аутентифікаторів, робота на різних операційних системах і браузерах. Водночас інтеграція з сучасними веб-додатками дозволяє зберігати стабільність навіть при масштабуванні, адже серверна частина ефективно обробляє дані та виконує перевірки без значних затримок.

WebAuthn поєднує високу юзабіліті з швидкодією та стабільністю, забезпечуючи безпечну та комфортну аутентифікацію для широкого спектру користувачів і сценаріїв.

4.3 Перспективи розвитку WebAuthn

WebAuthn та Passkeys представляють собою сучасний підхід до безпарольної аутентифікації, який поступово змінює типові методи входу у веб-сервіси та мобільні додатки. WebAuthn, як стандарт W3C і FIDO Alliance, забезпечує безпечну взаємодію між користувачем та

сервером за допомогою криптографічних ключів. Ця технологія значно підвищує безпеку, оскільки унеможливлює типові атаки на паролі. З практичного боку це означає, що користувачі можуть швидко входити в різні сервіси, а адміністраторам не потрібно турбуватися про безпечне зберігання та управління паролями.

WebAuthn має значний потенціал для широкого розповсюдження в найближчі роки. Очікується, що вони стануть стандартом для корпоративних систем, державних сервісів та великих онлайн-платформ завдяки своїй безпеці. Розробники продовжують працювати над покращенням сумісності між різними пристроями та операційними системами, а також над інтеграцією з існуючими протоколами автентифікації. У майбутньому безпарольні технології можуть змінити паролі повністю. Що дозволить створити єдину та ще більш безпечну систему безпарольної аутентифікації для всіх типів користувачів та сервісів.

4.4 Перспективи використання WebAuthn у державних та комерційних системах

WebAuthn відкриває нові можливості для державних та комерційних систем завдяки своїй здатності забезпечувати високий рівень безпеки при простому і зручному для користувача процесі аутентифікації. У державних системах WebAuthn може стати основою для надійної електронної ідентифікації громадян. Це актуально для онлайн-платформ надання державних послуг, систем електронного голосування, податкових сервісів та медичних реєстрів. Використання криптографічних ключів замість паролів значно знижує ризик компрометації даних і підвищує довіру користувачів до державних сервісів. У комерційному секторі WebAuthn дозволяє підвищити безпеку фінансових транзакцій зменшуючи витрати на підтримку та відновлення паролів. Користувачі отримують можливість швидкого та безпечного

входу, а компанії отримують зниження ризику шахрайства та втрати репутації. Перевагою є також можливість інтеграції з існуючими рішеннями для багатофакторної аутентифікації.

Перспективи розвитку WebAuthn у державних та комерційних системах виглядають дуже обнадійливими. Зі зростанням прийняття безпарольних технологій, WebAuthn може стати стандартом як для масових користувачів так і для критичних систем, де безпека є ключовою. В майбутньому можливе посилення підтримки різних апаратних аутентифікаторів, покращення крос-платформеної сумісності та інтеграція з різними пристроями. Це дозволить створити єдину безпечну екосистему аутентифікації для державних і комерційних сервісів у світі.

4.5 Потенційні проблеми масового впровадження та шляхи їх вирішення

Масове впровадження WebAuthn стикається з кількома потенційними проблемами, які можуть гальмувати його поширення. Незважаючи на очевидні переваги в безпеці та зручності. Однією з головних складностей є апаратна залежність. Користувачі повинні мати відповідні аутентифікатори, будь то вбудовані платформи або зовнішні ключі. Це створює бар’єр для тих, хто не має сучасних пристроїв або фінансової можливості придбати їх. Цю проблему можна частково вирішити через масштабну підтримку крос-платформових аутентифікаторів і інтеграцію з мобільними пристроями.

Іншою проблемою є навчання користувачів та зміна звичних моделей аутентифікації. Люди в більшості звикли до паролів. Тому перехід на безпарольну систему потребує чітких інструкцій, інтерфейсів, що пояснюють процес. Також наявності механізмів відновлення доступу у випадку втрати ключа. Вирішення полягає в розробці інтуїтивно

зрозумілих UX-рішень. Крім цього, автоматизованих процедур відновлення та активній інформаційній кампанії для користувачів.

Інтеграція WebAuthn у великі державні та комерційні системи може зустрітися з технічними та організаційними труднощами. Необхідно забезпечити сумісність з існуючими API, SSO-провайдерами та корпоративними політиками безпеки. Вирішення цих проблем можливе через поетапне впровадження, тестування у контрольованому середовищі та поступове оновлення серверної інфраструктури. Питання стандартизації та підтримки різних браузерів і ОС залишається актуальним. Щоб уникнути фрагментації, потрібно підтримувати широке коло платформ і регулярно оновлювати сумісність із новими версіями браузерів та операційних систем. Додатково спрощуючи інтеграцію для розробників. В підсумку, подолання цих викликів дозволить масово впровадити WebAuthn, зробивши його ефективним і надійним стандартом безпарольної аутентифікації у державних і комерційних системах.

Висновки до розділу 4

Ймовірно що WebAuthn є однією з найперспективніших технологій безпарольної аутентифікації. У порівнянні з іншими безпарольними методами, такими як одноразові паролі, TOTP або апаратні токени. WebAuthn вирізняється особливостями захисту та уніфікованою підтримкою різних платформ і браузерів. На відміну від біометричних систем, WebAuthn забезпечує стандартизовану та контрольовану розробником взаємодію.

Оцінка юзабіліті, швидкодії та стабільності показує, що WebAuthn надає інтуїтивний та комфортний досвід для користувача. Стабільність досягається завдяки стандартизованій архітектурі та підтримці різних типів аутентифікаторів. Це дозволяє працювати з декількома пристроями і масштабувати систему без втрати ефективності. Використання

WebAuthn у державних та комерційних системах може підвищити безпеку критичних сервісів, таких як електронна ідентифікація громадян чи податкові сервіси. Інтеграція з існуючими протоколами, SSO та корпоративними політиками дозволяє створити єдину екосистему аутентифікації.

Масове впровадження WebAuthn може зустріти проблеми, пов’язані з апаратною залежністю. Також необхідністю навчання користувачів. Вирішення цієї проблеми передбачає поетапне впровадження, розробку інтуїтивного UX, впровадження автоматизованих процедури відновлення доступу. Разом з регулярним оновленням сумісності з браузерами та операційними системами.

ВИСНОВКИ

Проведене дослідження підтвердило високу актуальність інтеграції технології WebAuthn у веб-додатки як сучасного та надійного методу аутентифікації користувачів. Аналіз теоретичних основ автентифікації та класичних методів показав, що паролі та традиційні токени поступаються сучасним безпарольним рішенням з точки зору безпеки та зручності користувача. Технологія WebAuthn забезпечує ефективне шифрування ключів, захист від фішингу та MITM-атак. Також спрощує процес взаємодії користувача з веб-додатком через двосторонню криптографічну аутентифікацію.

У детально розглянуто архітектурні принципи інтеграції WebAuthn, механізми обробки даних на сервері та взаємодію з клієнтською частиною через JavaScript API. Описані процеси реєстрації та логіна, їх взаємодія з OAuth 2.0, OIDC та SSO-провайдерами. Вони демонструють можливість впровадження WebAuthn у сучасні корпоративні та комерційні системи аутентифікації. Проведений аналіз вразливостей та методів їх запобігання підкреслює необхідність врахування безпекових аспектів на всіх рівнях архітектури веб-додатків.

Розробка плану для інтеграції WebAuthn у веб-додатки показала, що застосування безпарольної аутентифікації підвищує рівень безпеки, оптимізує UX та створює можливості для масштабованого впровадження у різних сферах. Розроблена модель системи аутентифікації, що включає управління ключами, fallback-механізми та recovery, дозволяє мінімізувати ризики втрати доступу та забезпечує стабільну роботу веб-додатку.

Інтеграція WebAuthn у веб-додатки є доцільною та перспективною, оскільки поєднує високу безпеку, зручність користування та сумісність із сучасними стандартами управління ідентичністю. Використання даної технології сприяє підвищенню

загального рівня кібербезпеки, зменшенню ризиків компрометації облікових записів та створює основу для подальшого розвитку безпарольних систем аутентифікації. Хоча, на даний момент, для широкого розповсюдження такого способу аутентифікації знадобиться час, для поступового впровадження цієї технології. Оскільки парольні рішення, не дивлячись на свої недоліки, все ще залишаються популярним рішенням аутентифікації.

ПЕРЕЛІК ПОСИЛАНЬ

1. WebAuthn. *Yubico Developers*. URL: https://developers.yubico.com/WebAuthn/?utm_source=chatgpt.com (дата звернення: 8.11.2025).
2. Acar T., Belenkiy M., Küpçü A. Single password authentication. *Computer networks*. 2013. Т. 57, № 13. С. 2597–2614. URL: https://doi.org/10.1016/j.comnet.2013.05.007 (дата звернення: 8.11.2025).
3. Sandhu R. Password less authentication. *Indian journal of science and technology*. 2019. Т. 12, № 43. С. 1–6. URL: https://doi.org/10.17485/ijst/2019/v12i43/145555 (дата звернення: 9.11.2025).
4. Rivera-Dourado M., Gestal M., Vázquez-Naya J. Public-key Based Authentication with WebAuthn and FIDO. *MOL2NET'21, conference on molecular, biomedical & computational sciences and engineering, 7th ed.*, м. Sciforum.net, 25 січ. 2021 р. – 30 січ. 2022 р. Basel, Switzerland, 2021. URL: https://doi.org/10.3390/mol2net-07-11847 (дата звернення: 11.11.2025).
5. View of FIDO2 protocol and biometric security. *Periodica Journal of Modern Philosophy, Social Sciences and Humanities*. URL: https://periodica.org/index.php/journal/article/view/672/573 (дата звернення: 11.11.2025).
6. M M. S. FIDO2 passkey: the passwordless future. *International journal for research in applied science and engineering technology*. 2024. Т. 12, № 3. С. 2838–2847. URL: https://doi.org/10.22214/ijraset.2024.59504 (дата звернення: 12.11.2025).

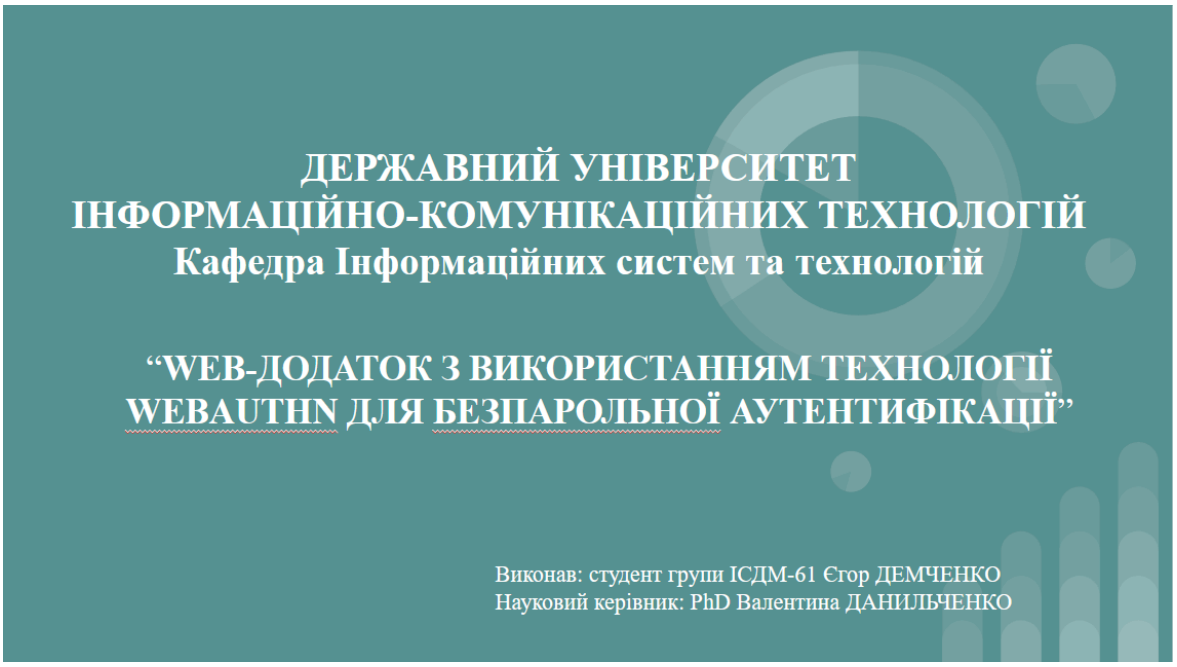
7. Defeating FIDO2/CTAP2/WebAuthn using browser in the middle and reflected cross site scripting / C. Catalano та ін. *Journal of Computer Virology and Hacking Techniques*. 2025. Т. 21, № 1. URL: https://doi.org/10.1007/s11416-025-00556-2 (дата звернення: 12.11.2025).
8. FIDO2 facing kleptographic threats by-design / M. Kutyłowski та ін. *Applied sciences*. 2024. Т. 14, № 23. С. 11371. URL: https://doi.org/10.3390/app142311371 (дата звернення: 14.11.2025).
9. Web authentication: an API for accessing public key credentials - level 2. *W3C*. URL: https://www.w3.org/TR/webauthn-2/?utm_source=chatgpt.com (дата звернення: 14.11.2025).
10. *arXiv.org e-Print archive*. URL: https://arxiv.org/pdf/2308.02973 (дата звернення: 17.11.2025).
11. *arXiv.org e-Print archive*. URL: https://arxiv.org/pdf/2412.02349 (дата звернення: 18.11.2025).
12. Authentication : thesis / Н. І. Муліна та ін. 2010. URL: http://essuir.sumdu.edu.ua/handle/123456789/17262 (дата звернення: 20.11.2025).
13. *CyberspACe securiTy and forensIcs lab (CactiLab)*. URL: https://cactilab.github.io/assets/pdf/jingjing2022fido2.pdf?utm_source=chatgpt.com (дата звернення: 20.11.2025).
14. Dourado M. R., Gestal M., Vázquez-Naya J. M. Implementing a web application for W3C webauthn protocol testing. *Proceedings*. 2020. Т. 54, № 1. С. 5. URL: https://doi.org/10.3390/proceedings2020054005 (дата звернення: 21.11.2025).
15. Maleh Y. Understanding web application security. *Web Application PenTesting*. New York, 2024. С. 39–69. URL:

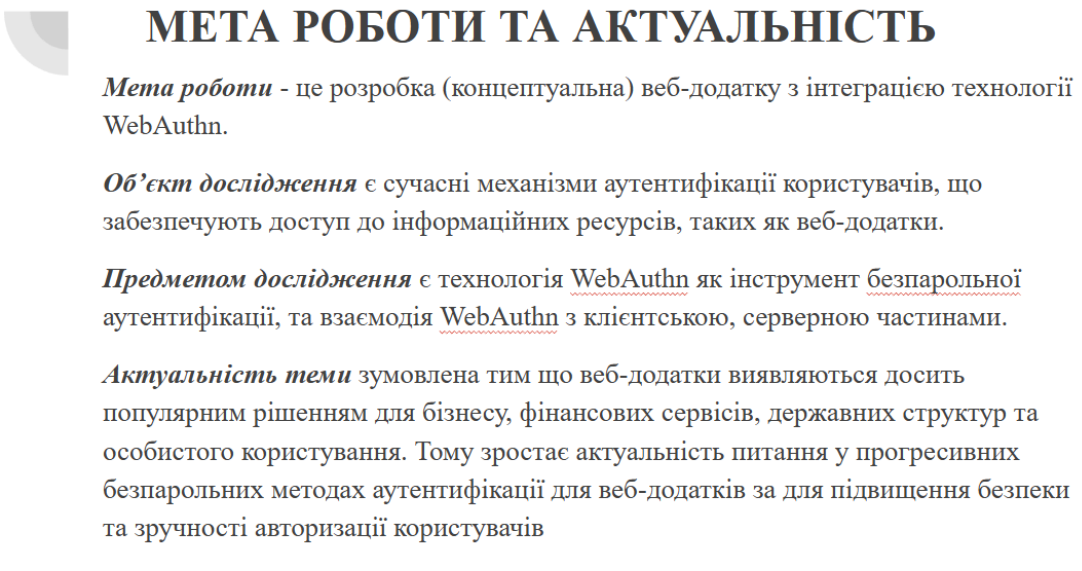
https://doi.org/10.1201/9788770046985-2 (дата звернення: 23.11.2025).

16. Marc R. WebAuthn - the future of web authentication. *Scip labs*. 2018. URL: https://doi.org/10.5281/zenodo.3521889 (дата звернення: 24.11.2025).
17. Mitra A., Ghosh A. FIDO2: a comprehensive study on passwordless authentication. *International journal of engineering research and applications*. 2024. Т. 14, № 7. С. 58–63. URL: https://doi.org/10.9790/9622-14075863 (дата звернення: 24.11.2025).
18. Nugent B. Password-based authentication. *ACM SIGSAC Review*. 1987. Т. 5, № 4. С. 10–13. URL: https://doi.org/10.1145/41818.41821 (дата звернення: 27.11.2025).
19. Panchenko A. R. Uses of webauthn protocol vulnerabilities to obtain unsanctioned access. *Voprosy kiberbezopasnosti*. 2025. Т. 6, № 70. С. 48–57. URL: https://doi.org/10.21681/2311-3456-2025-6-48-57 (дата звернення: 28.11.2025).
20. Researcher. Fido2: a new era in secure web authentication. *International journal of computer engineering and technology (IJCET)*. 2024. Т. 15, № 4. С. 841–858. URL: https://doi.org/10.5281/zenodo.13479154 (дата звернення: 30.11.2025).

Додаток А

КОПІЇ ДЕМОНСТРАЦІЙНИХ МАТЕРІАЛІВ





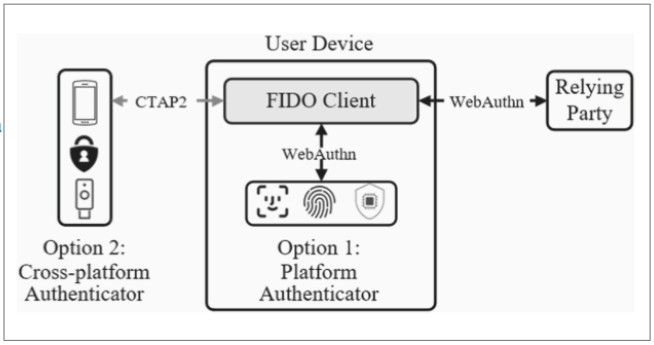
ЗАВДАННЯ РОБОТИ

1. Розглянути основні поняття аутентифікації, її роль у забезпеченні інформаційної безпеки та розглянути недоліки парольних методів аутентифікації.
2. Дослідити архітектуру WebAuthn.
3. Розробити концепт веб-додатку з інтеграцією WebAuthn, описати основні компоненти системи та основні алгоритми аутентифікації.
4. Оцінити ефективність та безпечність системи, порівняти її з іншими безпарольними технологіями.
5. Дослідити перспективи розвитку стандарту WebAuthn та можливості його впровадження у комерційних і державних веб-додатках.

АРХІТЕКТУРА БЕЗПАРОЛЬНОЇ АУТЕНТИФІКАЦІЇ У WEBAUTHN

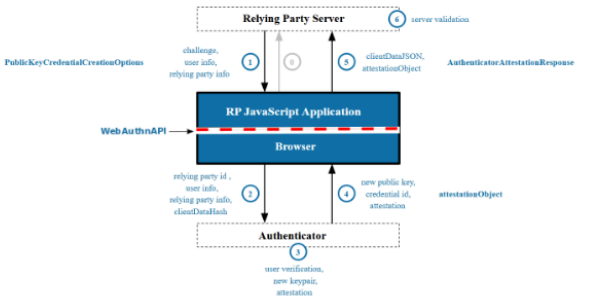
Стандарт FIDO2, він включає два основні компоненти:

- WebAuthn, який допомагає з взаємодією між веб-додатком та аутентифікатором користувача.
- CTAP, що забезпечує комунікацію між клієнтським пристроєм та аутентифікатором.



WEBAUTHN API: РЕЄСТРАЦІЯ

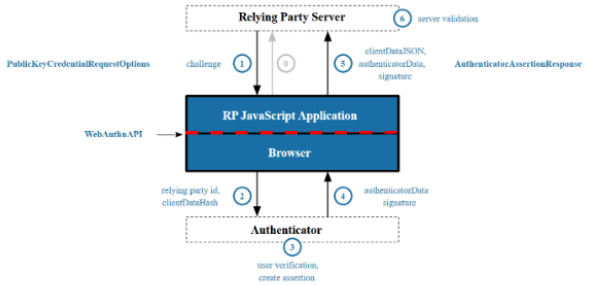
Цей розділ нормативно визначає API для створення та використання облікових даних відкритого ключа. Основна ідея полягає в тому, що облікові дані належать користувачеві та управляються WebAuthn Authenticator, з яким WebAuthn Relying Party взаємодіє через клієнтську платформу. Скрипти Relying Party можуть (за згодою користувача) вимагати від браузера створити нові облікові дані для майбутнього використання Relying Party.



WEBAUTHN API: АУТЕНТИФІКАЦІЯ

Скрипти також можуть запитувати дозвіл користувача на виконання операцій автентифікації з використанням існуючих облікових даних.

Всі такі операції виконуються в аутентифікаторі і опосередковуються клієнтською платформою від імені користувача. Скрипт ні в якому разі не отримує доступ до самих облікових даних; він отримує лише інформацію про облікові дані у вигляді об'єктів.



ЕТАПИ АУТЕНТИФІКАЦІЇ У WEBAUTHN

1. Користувач відвідує сайт, який виконує скрипт.
2. Скрипт запитує у клієнта підтвердження аутентифікації, надаючи якомога більше інформації, щоб звузити вибір прийнятних облікових даних для користувача. Це можна отримати з даних, які були збережені локально після реєстрації або запитуючи у користувача зареєстроване ім'я.
3. Скрипт довіреної сторони запускає код для аутентифікації.
4. Клієнтська платформа шукає та знаходить аутентифікатор.
5. Клієнт підключається до аутентифікатора.
6. При відкритті повідомлення користувачеві відкривається меню вибору прийнятних облікових даних. Для цього використовується інформація про обліковий запис, яка була надана при створенні облікових даних.
7. Аутентифікатор отримує від користувача біометричний або інші дані для авторизації.
8. Аутентифікатор повертає відповідь користувачу, а користувач повертає відповідь скрипту довіреної сторони. Якщо користувач відмовився вибрати обліковий запис або надати авторизацію, то повертається відповідна помилка.

РЕАЛІЗАЦІЯ АУТЕНТИФІКАЦІЇ НА ПРАКТИЦІ

Приклад реалізації аутентифікації, якщо скрипт довіреної сторони не має жодних локально збережених даних тощо.

```
if (!window.PublicKeyCredential) { /* Client not capable. Handle error. */ }

// credentialId is generated by the authenticator and is an opaque random byte array
var credentialId = new Uint8Array([183, 148, 245 /* more random bytes previously generated
var options = {
  // The challenge is produced by the server; see the Security Considerations
  challenge: new Uint8Array([4,101,15 /* 29 more random bytes generated by the server */]),
  timeout: 120000,  // 2 minutes
  allowCredentials: [{ type: "public-key", id: credentialId }]
};

navigator.credentials.get({ "publicKey": options })
    .then(function (assertion) {
    // Send assertion to server for verification
}).catch(function (err) {
    // No acceptable credential or user refused consent. Handle appropriately.
});
```

РЕАЛІЗАЦІЯ АУТЕНТИФІКАЦІЇ НА ПРАКТИЦІ

Якщо скрипт довіреної сторони містить локально збережені дані, які допомагають звузити список пошуку облікових даних. Цей приклад реалізації також демонструє, як використовувати розширення Credential Properties Extension (credProps).

```
if (!window.PublicKeyCredential) { /* Client not capable. Handle error. */ }

var encoder = new TextEncoder();
var acceptableCredential1 = {
    type: "public-key",
    id: encoder.encode("BA44712732CE")
};
var acceptableCredential2 = {
    type: "public-key",
    id: encoder.encode("BG35122345NF")
};

var options = {
  // The challenge is produced by the server; see the Security Considerations
  challenge: new Uint8Array([8,18,33 /* 29 more random bytes generated by the server */]),
  timeout: 120000,  // 2 minutes
  allowCredentials: [acceptableCredential1, acceptableCredential2],
  extensions: { 'credProps': true }
};

navigator.credentials.get({ "publicKey": options })
    .then(function (assertion) {
    // Send assertion to server for verification
}).catch(function (err) {
    // No acceptable credential or user refused consent. Handle appropriately.
});
```

\

ПОРІВНЯННЯ МЕТОДІВ БЕЗПАРОЛЬНОЇ АУТЕНТИФІКАЦІЇ

Метод аутентифікації	Переваги	Недоліки
Аутентифікація через електронну пошту	Має просту реалізацію, не потребує додаткової архітектури для впровадження.	Не гарантує високий рівень безпеки. Вразлива до фішингу. Найбільш вразливий метод безпарольної аутентифікації.
Багатофакторна автентифікація (MFA)	Забезпечує підвищений рівень безпеки, зменшує ризик компрометації облікового запису.	Потребує додаткової інфраструктури. Може погіршувати досвід користувача, оскільки вимагає введення додаткових кодів або підтверджень.
Біометрична автентифікація	Проста у використанні. Ускладнює підробку ключів для аутентифікації.	Вимагає спеціального обладнання та інфраструктури. Наприклад сканери відбитків пальців чи камери для розпізнавання обличчя.
Апаратні токени	Високий рівень безпеки, стійкість до фішингу. Ключі фізично захищені від атак поки не контактують з пристроєм.	Потребують фізичного носія. Можливий ризик втрати або пошкодження токена.
Одноразові коди (OTP)	Простота використання. Доступні великому обсягу користувачів. Можуть бути інтегровані з мобільними додатками.	Мають ризик перехоплення під час передачі. Менша надійність у порівнянні з іншими методами.

ВИСНОВКИ

1. Були розглянуті основні поняття аутентифікації, її роль у забезпеченні інформаційної безпеки та недоліки парольних методів аутентифікації.
2. Досліджена архітектура WebAuthn.
3. Розроблений концепт веб-додатку з інтеграцією WebAuthn, описані основні компоненти системи та основні алгоритми аутентифікації.
4. Оцінена ефективність та безпечність, порівняно з іншими безпарольними технологіями.
5. Досліджено перспективи розвитку стандарту WebAuthn та можливості його впровадження у комерційних і державних веб-додатках.

АПРОБАЦІЯ:
1. III Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу», на тему «Інтеграція технології автентифікації WebAuthn у web-додатках», що відбудеться 18 листопада 2025 року.

ДЯКУЮ ЗА УВАГУ!