КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра комп'ютерної інженерії

До захисту допущено: «На правах рукопису»
Завідувач кафедри _____ Юрій Бойко
« _ » _____ 2023 р.

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«ВИКОРИСТАННЯ ТЕХНОЛОГІЇ BLOCKCHAIN ДЛЯ АВТЕНТИФІКАЦІЇ У SPAN МЕРЕЖАХ»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 123 Комп'ютерна інженерія
ОНП «_____»
Богдан Карлаш _____

Науковий керівник:
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович _____

Рецензент:
_____

Засвідчую, що у цій магістерській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____

Робота допущена до захисту в ЕК рішенням кафедри _____
від «__»_____ 2023 р., протокол № __.

Завідувач кафедри _____,
кандидат фізико-математичних наук, доцент
Бойко Юрій Володимирович

(підпис)

Київ – 2023

РЕФЕРАТ

Дипломна робота магістра: 67 с., 22 рис., 2 табл., 52 джерела, 6 додатків.

В роботі виконано порівняльний аналіз програмних реалізацій спеціальних мереж смартфонів (SPAN), а саме FireChat, Bridgefy, Serval Project та Briar та проаналізовано їхні підходи до безпеки та автентифікації користувачів. В роботі виявлено, що існуючі механізми автентифікації користувачів недостатні та вразливі, тому запропоновано рішення, що використовує технологію розподілених застосунків Blockchain для автентифікації користувачів. Рішення написане на мові програмування Java та NextJS та використовує Ngrok для публічного доступу до застосунку з використанням SSL. Таке рішення використовує Web3 гаманець MetaMask для виконання процедури автентифікації та спеціальний API від Moralis. Результуюче рішення є більш безпечне завдяки особливостям роботи технології Blockchain та окрім автентифікації, дозволяє перевірити користувача в мережі Blockchain. Воно може бути застосоване для програмних реалізацій SPAN, а також для будь-яких інших мобільних застосунків, що потребують механізму автентифікації користувачів.

БЛОКЧЕЙН, СПЕЦІАЛЬНІ МЕРЕЖІ СМАРТФОНІВ, METAMASK, MORALIS, WEB3, SPAN.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ОС – Операційна Система

ПК – Персональний Комп’ютер

API – Application Programming Interface

Bluetooth LE – Bluetooth Low Energy

CC – Creative Commons

GNU GPL – GNU General Public License

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

ID – Identifier

IoT – Internet of Things

JWT – JSON Web Token

JSON – JavaScript Object Notation

MANET – Mobile Ad Hoc Network

QR – Quick Response

Tor – The Onion Router

SDK – Software Development Kit

SPAN – Smartphone Ad Hoc Network

SSL – Secure Sockets Layer

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

VANET – Vehicular Ad Hoc Network

WANET – Wireless Ad Hoc Network

ВСТУП

Наразі через неконтрольовані ситуації, що спричиняють тимчасове відімкнення електроенергії, такі як стихійні лиха, чи військові дії, чи аварії на розподільчих станціях, виникає необхідність в пошуку методів спілкування, що можуть працювати як за наявності мережі Інтернет, так і без доступу до неї. Існує декілька підходів до вирішення цього питання.

Першим рішенням є використовувати мобільний зв’язок для спілкування за допомогою дзвінків або спілкування в звичайних месенджерах у мережі Інтернет використовуючи мобільні дані. Однак проблема такого рішення полягає в тому, що за описаних ситуацій зі втратою або порушенням електропостачання, також порушується і сигнал мобільного зв’язку, коли не те, що використовувати мобільні дані для доступу в мережу Інтернет неможливо, навіть сигнал дзвінка може не доходити зовні в зону з проблемами з електропостачанням або навпаки.

Другим можливим рішенням є використання оптоволоконних кабелів, що більш стійкі до подібних ситуацій і можуть надавати зв’язок з мережею Інтернет навіть у випадку локальних проблем з електропостачанням, якщо в постачальника послуг є електроживлення. Однак, проблемою такого рішення є те, що такої опції, особливо в місці може не надаватись постачальником Інтернет послуг, а якщо така послуга і є, вона зазвичай коштує дорожче звичайного кабельного з’єднання.

Третім рішенням є встановлення системи безперебійного живлення на розподільчій станції, однак зазвичай провайдер цю опцію не оплачує і потрібна згода всіх в будинку.

Зрештою, є можливість використання програмних застосунків месенджерів, що реалізують так звані SPAN (Smartphone Ad Hoc Network) мережі, які здатні працювати як за наявності мережі Інтернет, так і без неї, а також не потребують існування попередньо налаштованої інфраструктури.

Враховуючи нещодавні часті ситуації з відключеннями електропостачання, спричинені військовими діями, це рішення стало актуальним та найбільш економічно вигідним, оскільки такі застосунки є безоплатні, однак вони мають недоліки із забезпеченням автентифікації користувачів.

В даній роботі пропонується розглянути та проаналізувати програмні реалізації SPAN та технологію Blockchain, що є децентралізованою технологією, яка активно розвивається та може використовуватись не лише для фінансів, що є популярним уявленням про неї, а й є корисним інструментом безпеки та може використовуватись і для автентифікації.

Мета роботи: запропонувати рішення з використанням технології Blockchain для автентифікації у SPAN мережах.

Розділ 1. Розподілені бездротові спеціальні мережі WANET

Бездротова ad hoc (спеціальна) мережа (WANET) – це сукупність двох або більше пристроїв, оснащених бездротовим зв’язком і мережевими можливостями. Такі пристрої можуть обмінюватися даними з іншим вузлом, який знаходиться безпосередньо в їх радіусі дії, або з тим, що знаходиться за її межами. Для останнього сценарію проміжний вузол використовується для ретрансляції або пересилання повідомлень від джерела до пункту призначення.

Мережа WANET є самоорганізованою та адаптивною. Це означає, що сформована мережа може бути деформована на льоту без необхідності будь-якого адміністрування системи. Вузли WANET повинні мати можливість виявляти присутність інших і виконувати необхідне встановлення зв’язків, щоб дозволити зв’язок і обмін інформацією та послугами.

Оскільки бездротові пристрої у WANET мережах можуть приймати різні форми (наприклад, смартфон чи ноутбук), можливості таких пристроїв щодо обчислення, зберігання та зв’язку можуть значно відрізнятися. Такі пристрої повинні не лише виявляти наявність зв’язку з сусідніми вузлами, а й визначати їхній тип та відповідні атрибути. Оскільки WANET не покладається на жодні

фіксовані мережеві об’єкти, сама мережа по суті не має інфраструктури в звичному її розумінні. Немає потреби ні в стаціонарних базових радіостанціях, ні в дротах, ні в стаціонарних маршрутизаторах [1].

1.1 Типи WANET мереж. Спеціальні мережі смартфонів SPAN

Децентралізований характер WANET робить їх придатними для різноманітних застосувань, де неможливо покластися на центральні вузли, і можна покращити масштабованість мереж порівняно з керованими бездротовими мережами. До основних можна віднести наступні:

- **Мобільна спеціальна мережа (MANET)** – це мережа мобільних пристроїв різного типу, підключених без дротів, що може безперервно самостійно конфігуруватись (є самоконфігурованою), самостійно організовуватись (є самоорганізованою) та не потребує інфраструктури (є безінфраструктурною). Такі мережі іноді ще відомі як мережі "на льоту" або "спонтанні мережі". В залежності від літератури, такий тип можуть виділяти як окремий або ж часто MANET ототожнюють з WANET.

- **Автомобільні спеціальні мережі (VANET)** – це тип WANET мережі, в якій пристроями є транспортні засоби та придорожнє обладнання. Транспортні засоби використовують радіохвилі для зв’язку один з одним та з придорожніми станціями, створюючи динамічні мережі, які адаптуються до швидких змін топології. VANET мережі дозволяють реалізувати різноманітні застосування, такі як безпека дорожнього руху, навігація, інформація про трафік, електронне гальмування та інші.

- **Спеціальні мережі смартфонів (SPAN)** – це мережі, пристроями в яких є смартфони. Такі мережі будуються завдяки наявному апаратному та програмному забезпеченню, наявному в смартфонах, що дозволяє їм відігравати роль маршрутизаторів та ретрансляторів, подібно до звичайних інфраструктурних мереж. Такі мережі не залежать від стільникового зв’язку та бездротових точок доступу, як-от домашніх

маршрутизаторів, натомість використовуються такі середовища обміну даними, як Bluetooth чи Wi-Fi Direct, або ж смартфони самі відіграють роль точки доступу.

- **Бездротові сітчасті (mesh) мережі** – це мережі, в яких кожен вузол приймає участь в передаванні даних та виступає в ролі комутатора, маршрутизатора чи ретранслятора для інших вузлів. Такі мережі мають більше з'єднань між пристроями, ніж звичайні топології мереж, як-от мережі типу «зірка», що дозволяє більшу відмовостійкість за рахунок підвищеної децентралізації. У повністю з'єднаній мережі кожен вузол з'єднаний з кожним іншим вузлом, утворюючи «сітку». Бездротові сітчасті мережі можуть бути побудовані на основі різних бездротових технологій, таких як Wi-Fi, WiMAX, Bluetooth або інших [2].

Відповідно до досліджень 2022 року від Statista та Ericsson, у 2021 році кількість власників смартфонів досягла майже 6,3 мільярди з 7,9 мільярдів населення всього світу [3], тобто майже 80% населення світу. Оцінюється, що в 2023 році ця кількість становитиме 6,8 – 6,9 мільярдів власників [4, 5].

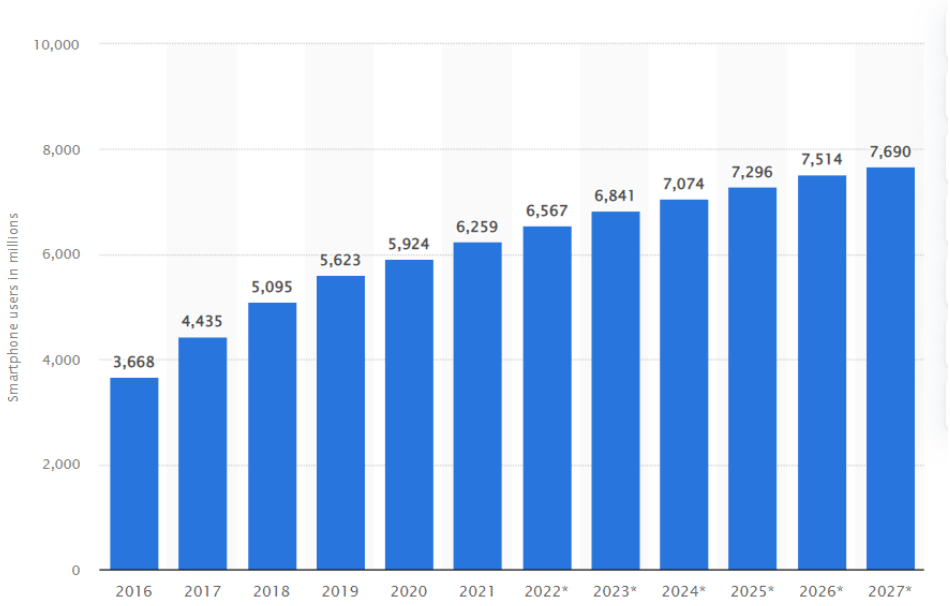

Рисунок 1 Statista – кількість власників смартфонів у 2016-2027 роках

Також, відповідно до Statista, кількість власників смартфонів в Україні станом на 2022 рік становила близько 33 мільйонів [6] з близько 41 мільйона населення України на 1 лютого 2022 року [7], що складає близько 80% населення країни. Це говорить про те, що переважна більшість людей є власниками смартфонів.

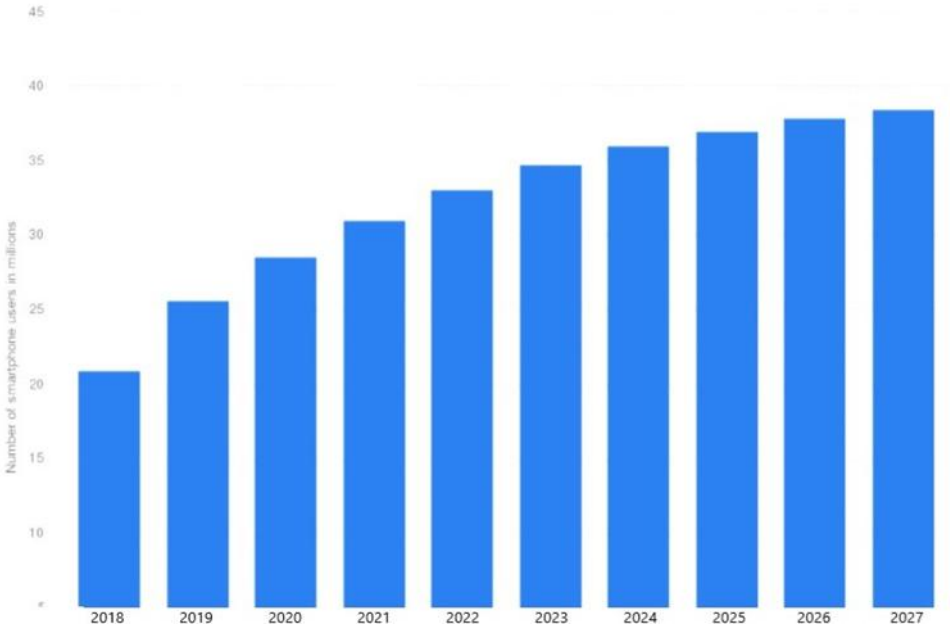

Рисунок 2 Statista – кількість власників смартфонів в Україні у 2018-2027 роках

Враховуючи це, пропонується розглянути SPAN мережі більш детально, оскільки вони не потребують ніякого спеціалізованого обладнання окрім смартфона або, в деяких випадках, ноутбука з підтримкою потрібного середовища передачі даних. SPAN мережі корисні за обставин, коли звичайна мережа перевантажена або недоступна, наприклад, на конференціях, музичних фестивалях, стихійних лихах чи при збройних конфліктах, а також вони здатні працювати як за наявності мережі Інтернет, так і без неї.

1.2 Порівняльний аналіз існуючих реалізацій SPAN: об’єкти та критерії аналізу

Пропонується проаналізувати існуючі реалізації SPAN, а саме **FireChat, Bridgefy, Serval Project та Briar**. Під час аналізу критеріями будуть: середовище передачі даних в реалізації, яка ліцензія використовується на

вихідний код, на яких платформах доступні застосунки, на якій мові програмування написана реалізація, де це можливо визначити відповідно до ліцензії, а також актуальність реалізації. Далі пропонується пояснити більш детально кожен критерій.

- **Середовище передачі даних.** Для розуміння можливого радіусу дії та швидкості, на якій передаватимуться дані варто проаналізувати яке середовище використовується. Загалом, це можуть бути такі середовища як Bluetooth чи Wi-Fi, проте можуть бути й інші.

- **Ліцензія на код.** Щоб розуміти, які обмеження накладаються на доступ до перегляду вихідного коду, його модифікацію та поширення, потрібно знати яку ліцензію встановили автори реалізації. Зазвичай ліцензія на код буває двох типів: пропрієтарна та відкрита. У випадку, коли ліцензія пропрієтарна, зазвичай доступу до перегляду чи модифікації вихідного коду не надається і він лишається комерційною таємницею, що є протилежністю до відкритих ліцензій на кшталт GNU GPL чи Creative Commons (CC).

- **Платформа.** Враховуючи особливість реалізацій, які піддаються аналізу, цією платформою має бути смартфон, однак варто також розділяти, що смартфони можуть бути на базі ОС Android чи iOS. Також, в окремих випадках, можуть також підтримуватись й такі пристрої як ноутбуки, які можуть бути на сімействах ОС Windows, macOS, чи Linux. В даній роботі пропонується прийняти для простоти термін «платформа» для визначення типу пристрою та ОС.

- **Мова програмування.** Зважаючи на використання різних платформ, можуть також використовуватися різні мови програмування для написання вихідного коду реалізації. Знання того, на якій мові програмування написано реалізацію, може бути корисним для більш детального розуміння як він працює, чи якщо розглядається можливість

використання вихідного коду в якійсь іншій реалізації. Оскільки зазвичай цей критерій напряму пов’язаний з ліцензією на код, коли немає можливості отримати до нього доступ пропонується позначати цей факт як n/a (недоступно).

- **Актуальність.** Цей критерій визначає, чи продовжується розробка реалізації та чи є до неї доступ. Для визначення цього можуть використовуватися різні підходи. Перш за все це доступність застосунку реалізації на офіційному майданчику відповідної платформи. Для Android це Google Play, також відомий як Play Store, а для iOS це App Store. Далі, якщо реалізація має відкриту ліцензію на вихідний код, можна в репозиторії бачити коли відбувалися останні зміни. Третім підходом може бути доступність застосунку на офіційному веб-сайті реалізації (поточна версія застосунку чи доступність самого веб-сайту).

1.3 Порівняльний аналіз існуючих реалізацій SPAN: аналіз реалізацій

1.3.1 FireChat

FireChat – це безкоштовна peer-to-peer програма для обміну повідомленнями, тобто надсилання тексту і зображень, яка працювала з доступом до Інтернету чи без нього [8]. FireChat розроблено компанією Open Garden та мав пропрієтарну ліцензію, тому деяку інформацію, як-от про мову програмування отримати неможливо. У лютому-березні 2020 року офіційна сторінка FireChat перестала існувати та з тих пір видає помилку 404 [9]. Наразі сторінка розробника видає помилку 403 [10].

Відповідно до архіву офіційного веб-сайту, FireChat використовував в якості середовища передачі Bluetooth та Wi-Fi для роботи без доступу до мережі Інтернет. Пристрої поруч підключалися між собою і працювали як ретранслятори, що дозволяло приєднувати пристрої в радіусі дії таких ретрансляторів, утворюючи меш мережу в межах групи до 50 людей [11]. Для

цього пристрої повинні були мати встановлену програму FireChat та знаходитись в радіусі 60 метрів.

FireChat був доступний на двох платформах: Android та iOS та його можна було встановити на офіційних майданчиках Google Play та App Store, відповідно. Станом на 2016 рік програма вже мала більше одного мільйону інсталяцій в Google Play [12].

1.3.2 Bridgefy

Bridgefy – це безкоштовний додаток для обміну повідомленнями, який працює без Інтернету. Bridgefy розроблено однойменною компанією та має пропрієтарну ліцензію, тому деяку інформацію, як-от про мову програмування отримати неможливо.

Відповідно до офіційного веб-сайту, програма використовує в якості середовища Bluetooth Low-Energy для під'єднання пристроїв в радіусі 100 м. Подібно до FireChat, Bridgefy також дозволяє працювати як ретранслятор та по ланцюгу передавати інформацію в меш мережі пристроїв [13]. Додаток Bridgefy працює завдяки Bridgefy SDK, яке є програмним забезпеченням, що дозволяє будь-якому мобільному додатку працювати без Інтернету. Розробникам пропонують використовувати даний SDK для забезпечення підтримки роботи власних застосунків всередині мереж Bridgefy [13].

Bridgefy також доступний на двох платформах: Android та iOS. Його можна встановити на офіційних майданчиках Google Play та App Store, відповідно. Програма має більше одного мільйону інсталяцій в Google Play [14]. Відповідно до офіційного веб-сайту, 8.3 мільйони користувачів користуються Bridgefy.

1.3.3 Serval Project

Serval Project, або просто Serval – телекомунікаційна система, що складається принаймні з двох мобільних телефонів, які можуть працювати за межами діапазону звичайних веж мобільного зв'язку завдяки спеціальному

застосунку [15]. Цим застосунком спочатку був так званий Serval Mesh (згодом перейменований на Batphone), що працює на платформі Android. Паралельно з ним розроблявся додаток для iOS Serval Chat, проте його вирішено переробити під Android та згодом ним замінили Serval Mesh [15, 16, 17].

Serval Mesh (також називається «Batphone») — це програма для Android 2.2 «Froyo» і вище. Він забезпечує безкоштовні безпечні голосові дзвінки з телефону на телефон, SMS і обмін файлами відбувається через Wi-Fi без необхідності використання SIM-карти чи комерційного оператора мобільного зв'язку [18].

Застосунок був доступний на майданчику Google Play до 2018 року і мав більше ста тисяч інсталяцій [19]. Наразі його можна завантажити лише з офіційного веб-сайту як застарілий застосунок.

Serval DNA є основним компонентом програми Serval Mesh для Android. Це демон, який виконує всі центральні служби меш мережі Serval, такі як динамічна маршрутизація, шифрування та автентифікація, розповсюдження файлів, обмін повідомленнями та голосова телефонія. Будь-який пристрій із підключенням до Wi-Fi, на якому працює демон Serval DNA, може брати участь у меш мережі Serval. Для мобільних пристроїв Android це означає запуск Serval Mesh [20]. На відміну від Serval Mesh, робота над Serval DNA все ще продовжується (остання зміна в репозиторії проекту відбувалась в липні 2022 року)

Вихідний код Serval Mesh надається для загального доступу згідно з GNU General Public License версії 3. Компонент Serval DNA надається для загального доступу згідно з GNU General Public License версії 2. Уся технічна документація є ліцензованою для громадськості за ліцензією Creative Commons Attribution 4.0 International [18]. Serval Mesh та його компонент Serval DNA написані на C та Java.

Serval Chat – це застосунок для обміну текстовими повідомленнями та соціальна мережа для Android написана на мові програмування Java. Програма є більш сучасним аналогом Serval Mesh з переробленим користувацьким інтерфейсом з акцентом на те, щоб полегшити спілкування за допомогою текстових повідомлень [17]. Застосунок також розроблявся під iOS на мові програмування Swift, проте робота над цією платформою припинено в 2018 році [21], а останні зміни до Android версії застосунку були в 2020 році [17].

Наразі Serval Chat можна завантажити з офіційного веб-сайту лише під платформу Android 4.0 та вище. Підключення між пристроями може відбуватися за допомогою як Bluetooth так і Wi-Fi середовищ.

1.3.4 Briar

Застосунок Briar, відповідно до розробників, розроблений для активістів, журналістів та всіх, хто потребує безпечного, легкого та надійного способу спілкування. У застосунку Briar повідомлення синхронізуються безпосередньо між пристроями користувачів. Якщо Інтернет не працює, Briar може синхронізуватися через середовища передачі Bluetooth або Wi-Fi, якщо ж доступ до мережі Інтернет є, Briar може може для цього використовувати мережу Tor. Також можлива опція підключення за допомогою знімних носіїв, наприклад USB накопичувача [22].

Briar є безкоштовним програмним забезпеченням. Його можна поширювати та змінювати згідно з умовами GNU General Public License. Більша частина (майже 98%) коду реалізації написана на мові програмування Java, також частково використовується Kotlin [23].

Briar наразі доступний перш за все на платформі Android та в активній розробці перебуває версія для ПК на ОС Windows та Linux, а також планується підтримка macOS в майбутньому. Його можна встановити на офіційних майданчиках Google Play та App Store, відповідно. Програма має більше одного мільйону інсталяцій в Google Play [24]. ПК версія застосунку відрізняється від

мобільної перш за все тим, що в ній підключення відбувається виключно за допомогою Wi-Fi, функціонал Bluetooth та підтримка підключення за допомогою знімних носіїв наразі недоступні [25].

Відповідно, можемо створити наступну таблицю, що буде містити всі проаналізовані проєкти з відповідними критеріями:

Таблиця 1 – Аналіз реалізацій SPAN

Назва	**Середовище передачі**	**Ліцензія на код**	**Платформа**	**Мова програмування**	**Актуальність**
FireChat	Bluetooth, Wi-Fi	Пропрієтарна	Android та iOS	n/a	2020 рік, припинено
Bridgefy	Bluetooth LE	Пропрієтарна	Android та iOS	n/a	2023 рік
Serval Project	Bluetooth, Wi-Fi	GNU GPL	Android	Java, C	2018 - 2022 рік, в розробці
Briar	Bluetooth, Wi-Fi, USB	GNU GPL	Android; Windows та Linux (beta)	Java, Kotlin	2023 рік

Як видно з Таблиці 1, найбільш актуальними є такі реалізації як Bridgefy та Briar, на другому місці знаходиться Serval Project, що перебуває в процесі розробки. Також видно що співвідношення відкритої ліцензії на код (Serval Project та Briar) до пропрієтарної (FireChat та Bridgefy) між усіма проєктами виявилось рівноцінне і, варто зауважити, що саме пропрієтарні проєкти підтримують платформу iOS, що можна пояснити політикою Apple щодо відкритих ліцензій на майданчику App Store. Проте видно, що усі проєкти працюють (або у випадку FireChat працювали) на ОС Android, що пояснює вибір мови програмування Java чи Kotlin, оскільки це основні мови програмування для цієї платформи.

Якщо говорити про середовище передачі, видно, що практично усі використовують Bluetooth та Wi-Fi, за виключенням лише Bridgefy, який натомість використовує більш енергозберігаючий Bluetooth LE. Також

особливістю Briar є підтримка альтернативних середовищ, як-от USB. Ще одною переважаючою особливістю Briar є те, що він підтримує не лише мобільні пристрої, а й у розробці має підтримку також ПК на ОС Windows та Linux.

Відповідно, враховуючи все сказане, відповідно до встановлених критеріїв, найвищу позицію має Briar.

1.4 Аналіз підходів до безпеки в реалізаціях SPAN

Перейдемо тепер до аналізу підходів до безпеки в аналізованих реалізаціях SPAN. Як вже зазначено, офіційно отримати доступ до програмного забезпечення FireChat неможливо, тому в цій частині буде протестовано лише три застосунки з чотирьох реалізацій, а саме Briar, Bridgefy, та Serval Chat.

Для аналізу підходів до безпеки використано такі інструменти:

- Смартфон з ОС Android 11: Xiaomi Redmi Note 8 Pro,
- Смартфон з ОС Android 4.4: Samsung Galaxy Duos,
- Ноутбук з ОС Windows 10: Asus GL502VM,
- Емулятор пристрою Android: BlueStacks версії 5.11.40
- Briar версії 1.4.23,
- Bridgefy версії 3.1.18,
- Serval Chat версії alpha 0.01

Пропонується проаналізувати наступне: у який спосіб виконується створення нового облікового запису чи нового користувача (де можна створити більше одного), у який спосіб користувач може розпочати спілкування з іншим користувачем, а також які є унікальні підходи до безпеки.

1.4.1 Briar

Застосунок Briar забороняє робити знімки екрану на пристрої, а також не підтримує версії Android 4, тому для цього тесту знімки взяті виключно з емулятору.

Безпека в застосунку Briar має такі особливості:

1) Створення облікового запису в Briar передбачає вибір унікального імені користувача та пароля, які зберігаються локально. У разі, якщо користувач не зможе згадати пароль, відновлення паролю неможливе і обліковий запис може бути втрачено назавжди, що можна побачити на Рисунку 3.

Рисунок 3 Створення облікового запису в застосунку Briar

2) Спілкування в застосунку Briar можливе лише між контактами, які взаємно погодилися підключитися. Для цього потрібно або обмінятися посиланнями через мережу Інтернет (Рисунок 4а) , або відсканувати QR-

коди одне одного (Рисунок 4б), що можливо зробити лише знаходячись поблизу.

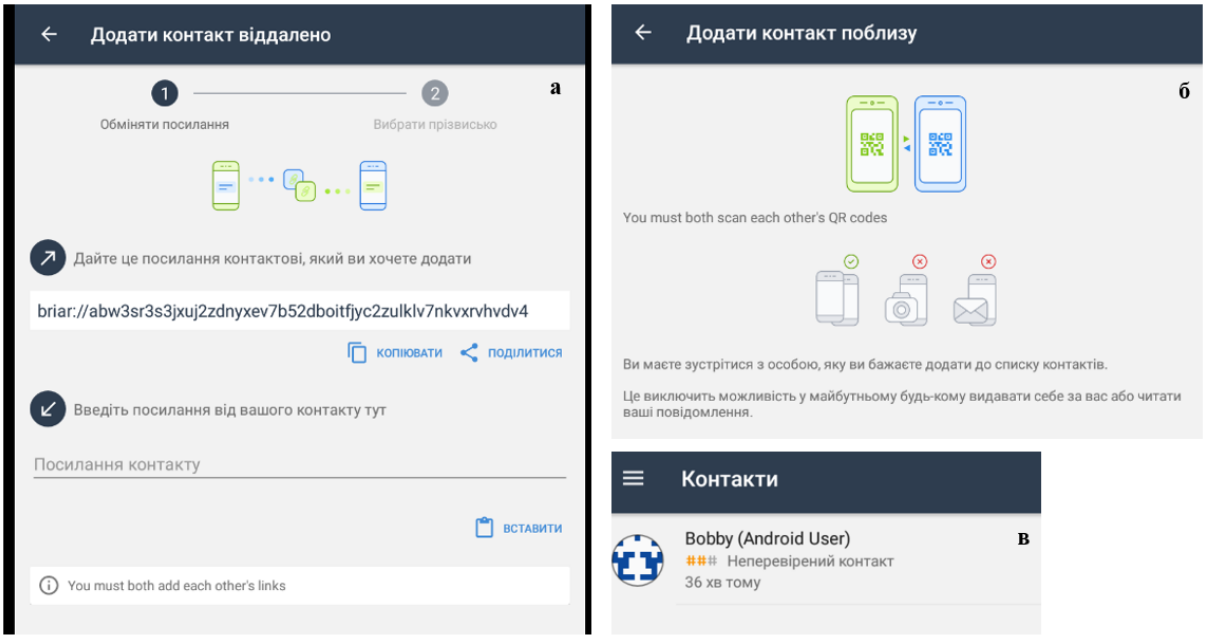

Рисунок 4 Додавання контактів у застосунку Briar.
а – додавання контакту віддалено, б – додавання контакту поблизу, в – доданий віддалено контакт.

3) Контакти можуть запрошувати один одного приєднатися до приватних каналів або форумів, але це може відбуватися лише за взаємною згодою, як видно на Рисунку 5.

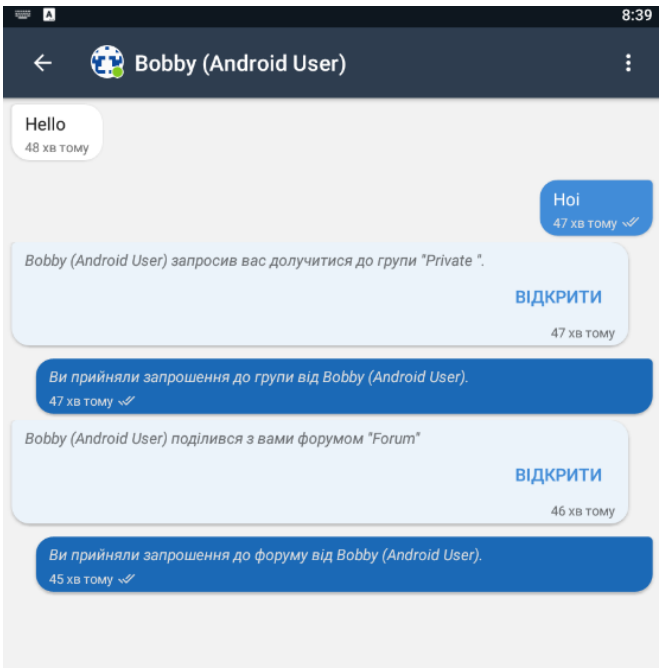

Рисунок 5 Запрошення на додавання до групи та форуму в застосунку Briar

4) У застосунку передбачена підтримка додатків тривожної кнопки, які, якщо встановлено, дозволяють або вийти з Briar, або видалити обліковий запис.

5) Під час встановлення підключення за допомогою мережі Інтернет Briar дозволяє використовувати TOR.

6) Briar позначає контакти, додані за допомогою обміну посиланнями, як «неперевірені», а додані через QR-коди як «перевірені». Таким чином, єдиний спосіб перевірити користувача – це сканувати його QR-код у безпосередній близькості.

7) Існує потенційна проблема з початковим з'єднанням при використанні обміну посиланнями для початку спілкування. У такому випадку неможливо підтвердити особу користувача, з яким починається спілкування, оскільки не існує ніякої процедури перевірки облікових записів при їхньому створенні.

Підсумовуючи, програма Briar вимагає взаємної згоди для спілкування та використовує QR-коди для підтвердження особи. Користувачі повинні ретельно

керувати своїми обліковими даними для входу, оскільки відновлення облікового запису неможливо. Додаток також пропонує додаткові заходи безпеки через TOR і функцію тривожної кнопки. Проблема перевірки облікового запису для контактів, доданих через обмін посиланнями, створює потенційні ризики для безпеки, які слід враховувати.

1.4.2 Bridgefy

Застосунок Bridgefy Використовує виключно Bluetooth, та не підтримує Android 4, тому повноцінна перевірка неможлива за допомогою наявних інструментів. Однак, особливості підходу до створення облікового запису та додавання контактів все ж перевірити можливо.

Створення облікового запису в Bridgefy вимагає лише імені та не передбачає використання паролів (Рисунок 6а).

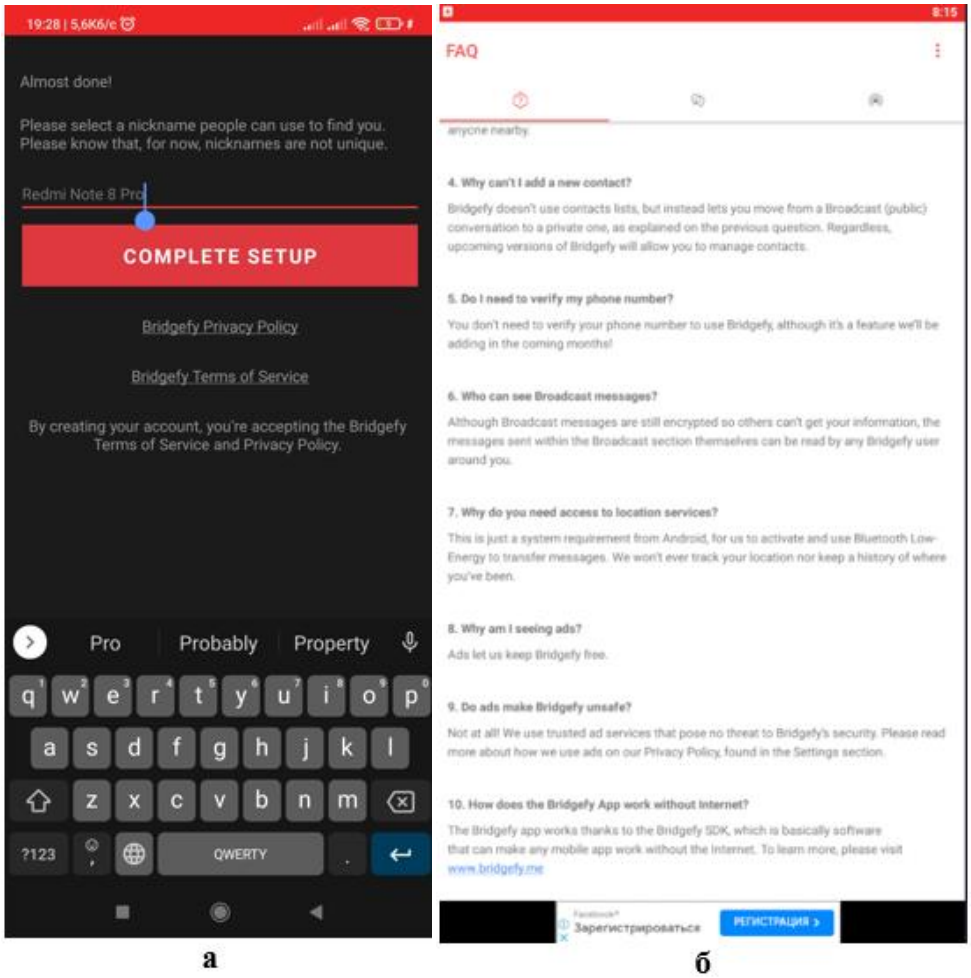

Рисунок 6 Створення облікового запису в застосунку Bridgefy (а), основна сторінка застосунку з рекламою (б)

Додавання контакту в Bridgefy передбачає трансляцію зашифрованого повідомлення-запрошення усім користувачам програми, що знаходяться поблизу (Рисунок 7). Переглянувши повідомлення, користувач може розпочати приватний чат із відправником. Також цей застосунок єдиний з перевірених, що показує рекламу (Рисунок 6б), яка може бути персоналізованою, що не є безпечною практикою.

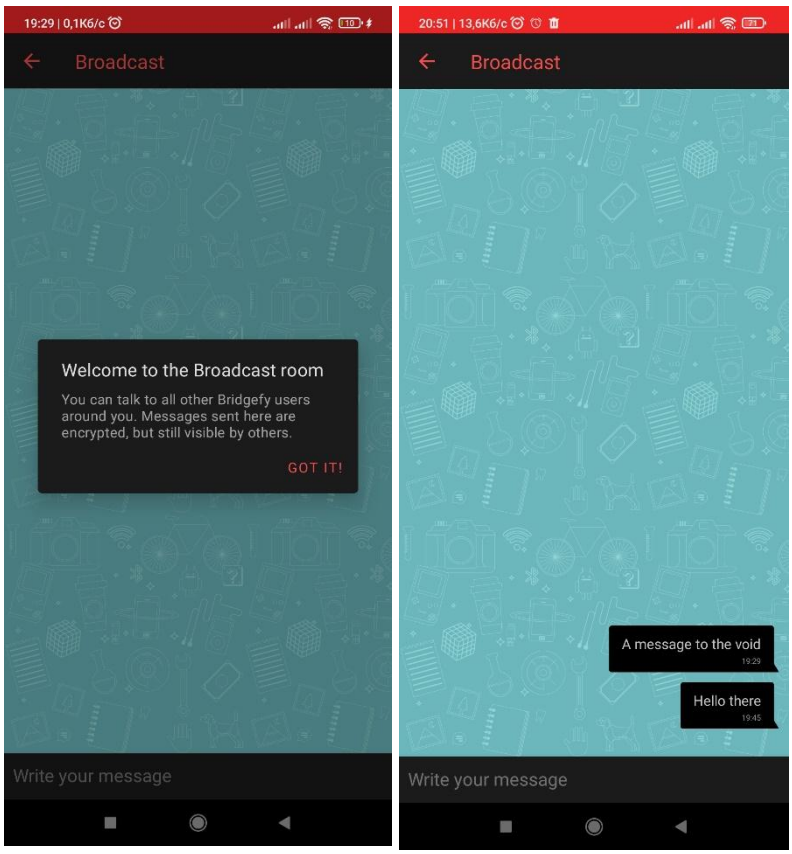

Рисунок 7 «Broadcast кімната» в Bridgefy для додавання контактів

Відповідно, можна бачити, що суттєвою є проблема відсутності необхідності створення паролю в застосунку. Також відсутні способи дистанційного додавання контактів, а наявна процедура передбачає надсилання публічного повідомлення і відсутня перевірка особи користувача.

1.4.3 Serval Chat

Застосунок Serval Chat підтримує версію Android 4, тому для його перевірки можливе використання усіх інструментів.

Особливості підходу до безпеки в застосунку Serval Chat є наступними:

1) Для створення облікового запису в Serval Chat не потрібно використовувати паролі. Користувачі можуть створити обліковий запис і розпочати обмін повідомленнями без будь-яких додаткових дій (Рисунок 8).

2) На одному пристрої можна створити кілька облікових записів, так званих ідентифікаторів (Рисунок 8), що ймовірно використовується для тестування роботи застосунку.

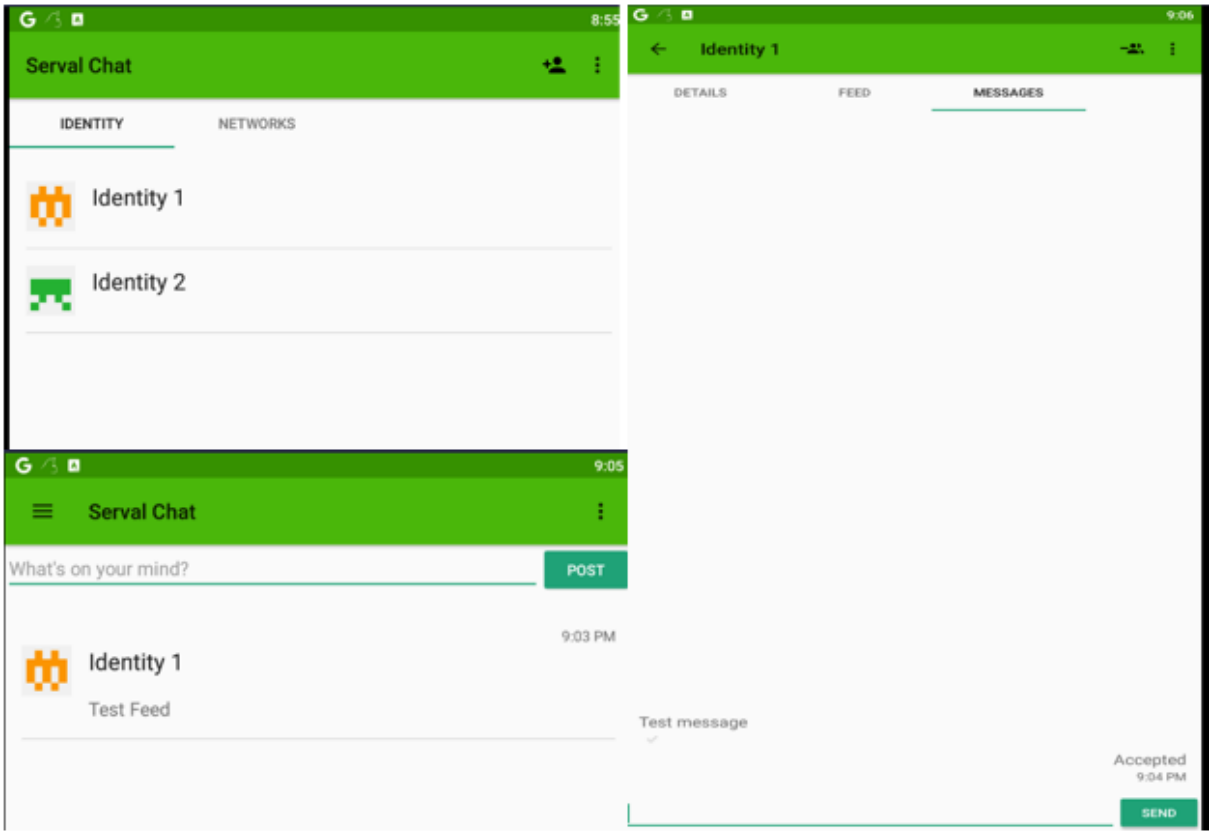

Рисунок 8 Створені облікові записи в застосунку Serval Chat та спілкування між користувачами

3) Програма підтримує функцію пошуку через Wi-Fi, Bluetooth або шляхом створення точки доступу, однак Serval Chat дозволяє встановлювати з’єднання лише між пристроями в одній мережі, тобто користувачі повинні користуватися однією точкою Wi-Fi або обидва повинні використовувати Bluetooth для підключення. Якщо один з користувачів знаходиться в мережі Wi-Fi, а інший використовує Bluetooth для додавання контактів, така спроба буде безуспішною.

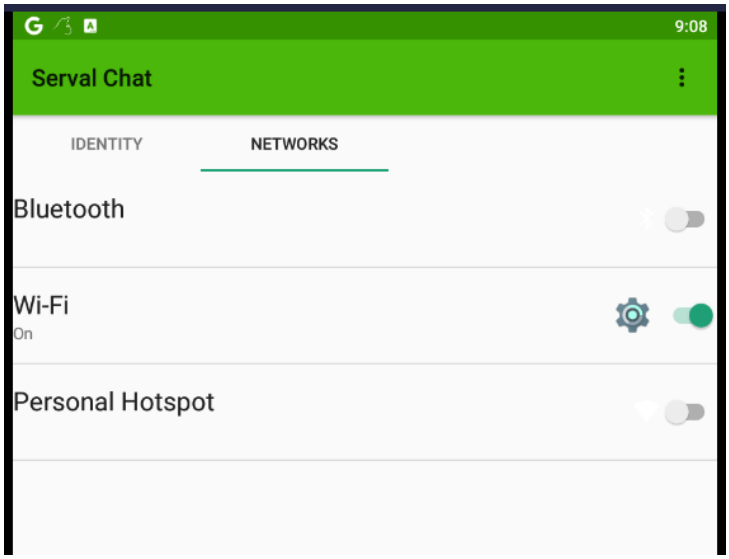

Рисунок 9 Способи підключення до інших користувачів в Serval Chat

Таким чином, програма Serval Chat пропонує спрощений процес створення облікового запису та підтримує різні параметри пошуку контактів. На одному пристрої можна створити кілька користувачів. Пристрої можуть підключатися один до одного виключно в одній мережі. Також в додатку немає ніякої перевірки особи користувача.

1.5 Висновки до розділу

Як можна бачити, в усіх проаналізованих застосунках є свої підходи до створення акаунтів та початку спілкування. Видно чітку градацію між проаналізованими застосунками:

- Serval Chat дозволяє без будь-яких попередніх запитів надсилати повідомлення кому завгодно в радіусі, та можна створити скільки завгодно акаунтів на одному пристрої.
- Bridgefy у свою чергу використовує підхід, що кожен може бачити повідомлення-запит на з’єднання, після чого може відбутися приватне спілкування, можна створити один акаунт, який не потребує паролю.
- Briar використовує підхід найбільше орієнтований на безпеку – почати спілкування можна лише з контактами, які можна встановити лише

обмінявшись посиланнями або відсканувавши QR-коди одне одного, крім того, щоб створити акаунт необхідне використання паролю.

Можна зробити висновок, що підхід, обраний Briar є найбільш безпечним, однак він все ж має недоліки. Перш за все, єдиний спосіб верифікувати, що користувач той, за кого він себе видає – це зустрітися з ним, що не завжди є оптимальним, особливо оскільки є можливість спілкування через мережу Інтернет на великих відстанях. По друге, не зважаючи на те, що створення облікового запису вимагає паролю, не існує ніякої верифікації та автентифікації особи, що може бути зловживано. Варто також зауважити, що описана проблема притаманна усім перевіреним застосункам.

Відповідно, доцільно проробити підхід, за допомогою якого можна вирішити названі проблеми з автентифікацією та верифікацією користувачів в децентралізованих застосунках – реалізаціях SPAN.

Розділ 2. Використання технології Blockchain для автентифікації у мережах SPAN

2.1 Можливі методи автентифікації користувачів в мережах SPAN

Як вже було розглянуто в розділі 1, реалізації SPAN мають проблеми автентифікації та верифікації користувачів, оскільки немає механізму, який би забезпечував перевірку ідентичності та надання дозволів. Механізми автентифікації мають вирішальне значення для безпеки та конфіденційності програм SPAN, оскільки вони запобігають несанкціонованому доступу та зловмисним атакам. Для SPAN можуть бути використані декілька методів автентифікації, наприклад:

- Методи на основі сертифікатів: ці методи покладаються на цифрові сертифікати, видані довіреним органом для автентифікації користувачів і пристроїв. Сертифікати містять відкриті ключі та ідентифікаційну інформацію, яку можуть перевірити інші сторони. Однак методи на основі сертифікатів вимагають використання довіреної третьої сторони

чи центру, що порушує підхід до децентралізації, який використовується в SPAN. Крім того, сертифікати можуть бути відкликані або прострочені, що вимагає частих оновлень і синхронізації.

- Методи на основі паролів: ці методи використовують паролі або PIN-коди для автентифікації користувачів і пристроїв. Паролі прості у використанні та не потребують додаткового апаратного чи програмного забезпечення. Однак методи, засновані на паролях, вразливі до багатьох кібератак. Крім того, паролі можуть бути забуті або втрачені, для чого потрібні механізми скидання пароля, яких зазвичай немає.

- Біометричні методи: ці методи використовують біометричні функції, такі як відбитки пальців, розпізнавання обличчя, розпізнавання райдужної оболонки ока або розпізнавання голосу для автентифікації користувачів і пристроїв. Біометричні характеристики унікальні, їх важко підробити чи вкрасти. Однак біометричні методи потребують спеціальних датчиків і пристроїв, які можуть бути недоступними або сумісними з усіма смартфонами. Крім того, біометричні характеристики можуть змінюватися з часом або на них можуть впливати навколишні фактори, що може знизити їх точність і надійність.

- Методи на основі технології Blockchain: ці методи використовують технологію Blockchain для створення розподіленого реєстру, що записує транзакції та події в SPAN. Blockchain – це однорангова мережа вузлів, які підтримують спільний і незмінний запис даних за допомогою криптографічних методів. Блокчейн може забезпечувати автентифікацію для SPAN, надаючи:

    - Управління ідентифікацією: Blockchain може децентралізовано та безпечно зберігати ідентифікаційну інформацію, таку як відкриті ключі, атрибути та облікові дані користувачів і пристроїв. Користувачі та пристрої можуть

реєструвати свої ідентифікаційні дані в блокчейні та оновлювати їх за потреби. Інші користувачі можуть підтвердити ідентичність, перевіривши записи блокчейну [26, 27].

- Контроль доступу: Blockchain може застосовувати політики контролю доступу на основі ідентифікаційної інформації, що зберігається в блокчейні. Користувачі та пристрої можуть запитувати доступ до ресурсів або послуг у SPAN, надсилаючи транзакції в блокчейн. Blockchain може перевіряти транзакції та надавати або забороняти доступ на основі попередньо визначених правил [26, 27].
- Можливість перевірки: Blockchain може забезпечити можливість перевірки автентифікації в SPAN, зберігаючи захищений від підробки та прозорий запис усіх транзакцій і подій у блокчейні. Користувачі та пристрої можуть переглядати історію блокчейну, щоб перевіряти, наприклад, хто отримував доступ до яких ресурсів або послуг, коли, де та як. Блокчейн також може виявляти та запобігати зловмисним діям за допомогою механізмів консенсусу та криптографічних доказів [26, 27].

На основі цього можна зробити висновок, що методи на основі Blockchain більше підходять для автентифікації в SPAN, ніж інші методи, оскільки вони пропонують більше переваг, таких як децентралізація, безпека та можливість перевірки. Методи, засновані на блокчейні, також можуть подолати деякі обмеження інших методів, такі як залежність від третіх сторін, низький рівень безпеки, висока вартість, проблеми сумісності, проблеми відкликання та закінчення терміну дії, проблеми керування паролями, проблеми точності та надійності. Однак методи, засновані на блокчейні, також мають деякі недоліки,

такі як високі обчислювальні витрати, обмеження пам’яті та проблеми з підключенням до мережі, які варто враховувати.

2.2 Особливості технології Blockchain

Пропонується більш детально розглянути особливості технології Blockchain загалом.

Блокчейн – це загальний незмінний реєстр, що полегшує процес запису транзакцій і відстеження активів у мережі. Актив може бути матеріальним або нематеріальним. Практично все, що має цінність, можна відстежувати в мережі блокчейн, що знижує ризики та скорочує витрати для всіх учасників [28]. База даних блокчейну зберігає дані в блоках, які з’єднані разом у ланцюг. Дані є хронологічно узгодженими, оскільки неможливо видалити або змінити ланцюжок без консенсусу в мережі [29].

За допомогою традиційних методів запису транзакцій і відстеження активів учасники мережі ведуть власні реєстри та інші записи. Цей традиційний підхід може бути дорогим, частково через те, що він передбачає використання посередників, які стягують плату за свої послуги. Це неефективно через затримки у виконанні угод і дублювання зусиль, необхідних для ведення численних реєстрів. Такий підхід також вразливий, тому що якщо центральна система (наприклад, банк, центральний сервер) скомпрометована через шахрайство, кібератаку або звичайну помилку, це може вплинути на всю мережу [28].

Технологія блокчейн має такі основні особливості як децентралізація, незмінність та консенсус:

- **Децентралізація** в блокчейні означає передачу контролю та прийняття рішень від централізованої сутності (індивіда, організації або групи) до розподіленої мережі. Децентралізовані блокчейн-мережі використовують прозорість, щоб зменшити потребу в довірі між учасниками. Ці мережі

також не дають учасникам здійснювати владу чи контроль один над одним у спосіб, що погіршує функціональність мережі.

- **Незмінність** означає, що щось не можна змінити. Жоден учасник не може втручатися в транзакцію, якщо хтось записав її до спільного реєстру. Якщо запис транзакції містить помилку, потрібно додати нову транзакцію, щоб скасувати помилку, і обидві транзакції будуть видимі в мережі.

- Відповідно до **консенсусу**, система блокчейн встановлює правила щодо згоди учасників на запис транзакцій. Учасник може фіксувати нові транзакції лише за умови згоди більшості в мережі.

В технології блокчейн також можна виділити три основні компоненти, завдяки яким вона працює – розподілений реєстр, смарт-контракти та криптографія з відкритим ключем.

- **Розподілений реєстр** – це спільна база даних у мережі блокчейн, яка зберігає транзакції, наприклад спільний файл, який може редагувати кожен у команді. У більшості спільних текстових редакторів (як-от Google Docs) будь-хто з правами редагування може видалити весь файл. Однак технології розподіленого реєстру мають суворі правила щодо того, хто може редагувати та як редагувати. Наприклад, учасник мережі не може видалити записи після того як їх було зафіксовано в мережі.

- **Смарт-контракти** – це програми, що зберігаються в системі блокчейн, які запускаються автоматично, коли виконуються заздалегідь визначені умови. Вони запускають перевірки «якщо-тоді», щоб завершувати транзакції. Наприклад, логістична компанія може мати смарт-контракт, який автоматично здійснює оплату, коли товар прибуває в порт.

- **Криптографія з відкритим ключем** – це функція безпеки для унікальної ідентифікації учасників мережі блокчейн. Цей механізм генерує два набори ключів для учасників мережі. Один ключ — це відкритий ключ,

спільний для всіх учасників мережі. Інший — закритий ключ, унікальний для кожного учасника. Приватний і відкритий ключі працюють разом, щоб розблокувати дані в реєстрі [29].

Своєю назвою блокчейн (blockchain) зобов'язаний тому, як він зберігає дані транзакцій – у блоках (block), які з'єднані між собою, щоб утворити ланцюг (chain) (Рисунок 10). Зі збільшенням кількості транзакцій зростає і блокчейн. Блоки записують і підтверджують час і послідовність транзакцій, які потім реєструються в блокчейні в межах окремої мережі, керованої правилами, погодженими учасниками мережі.

Рисунок 10 Збереження транзакцій в послідовно зв'язаному ланцюгові блоків

Кожен блок містить хеш, пакети останніх дійсних транзакцій із мітками часу та хеш попереднього блоку. Хеш попереднього блоку пов'язує блоки разом і запобігає будь-якій зміні до блоку або вставці нового блоку між двома існуючими. Таким чином, кожен наступний блок посилює перевірку попереднього блоку і, отже, всього блокчейну. Цей метод робить блокчейн захищеним від несанкціонованого доступу, надаючи ключовий атрибут незмінності.

Блокчейни можуть бути **закриті (permissioned)** з дозволами та **відкриті (permissionless)** без дозволів. В закритому блокчейні кожен учасник має унікальний ідентифікатор, що дозволяє використовувати політики для обмеження участі в мережі та доступу до деталей транзакцій. Маючи

можливість обмежувати участь у мережі, організації можуть легше дотримуватися правил захисту даних, таких як ті, що передбачені в HIPAA, GDPR, та інших. Закриті блокчейни також більш ефективні для контролю узгодженості даних, які додаються до блокчейну.

Завдяки можливості обмежити доступ до деталей транзакцій у блокчейні можна зберігати більше деталей транзакцій, а учасники можуть вказати інформацію про транзакції, яку вони бажають надати іншим. Крім того, деяким учасникам може бути дозволено переглядати лише певні транзакції, тоді як іншим, наприклад аудиторам, може бути надано доступ до більш широкого спектру транзакцій. З відкритим блокчейном, навпаки, рівень деталізації транзакцій може бути обмежений для захисту конфіденційності та забезпечення анонімності.

Наприклад, якщо сторона A передає актив стороні B, обидві сторони можуть бачити деталі транзакції. Сторона C бачить, що A і B провели транзакцію, але не може бачити подробиці передачі активів. Якщо аудитор або регулятор приєднується до мережі, служби конфіденційності можуть гарантувати, що тільки аудитор бачить повну інформацію про всі транзакції в мережі. Цього можна досягти за допомогою цифрових сертифікатів. Подібно до паспорта, цифровий сертифікат надає ідентифікаційну інформацію, є стійким до підробок і може бути перевірений, оскільки його видано перевіреною особою/організацією. Мережа блокчейну в такому випадку включає центр сертифікації, який і видає цифровий сертифікат [28].

Існує чотири основні типи децентралізованих або розподілених мереж у блокчейні: публічні, приватні, гібридні та консорціум блокчейн мережі.

- **Публічні**, або загальнодоступні, блокчейни зазвичай відкриті і дозволяють кожному приєднатися до них. Усі учасники блокчейну мають рівні права читати, редагувати та перевіряти блокчейн. Люди переважно

використовують загальнодоступні блокчейни для обміну та майнінгу криптовалют, таких як Bitcoin, Ethereum та Litecoin.

- Одна організація контролює **приватні** блокчейни, які також називають керованими блокчейнами. Дозволи визначають, хто може бути учасником і які права вони мають у мережі. Приватні блокчейни лише частково децентралізовані, оскільки вони мають обмеження доступу. Ripple, мережа обміну цифрової валюти для бізнесу, є прикладом приватного блокчейна.

- **Гібридні** блокчейни поєднують елементи як приватних, так і публічних мереж. Компанії можуть створювати приватні системи на основі дозволів поряд із загальнодоступною системою. Таким чином вони контролюють доступ до конкретних даних, що зберігаються в блокчейні, зберігаючи решту даних загальнодоступною. Вони використовують смарт-контракти, щоб дозволити публічним членам перевіряти, чи були завершені приватні транзакції. Наприклад, гібридні блокчейни можуть надати публічний доступ до цифрової валюти, зберігаючи банківську валюту приватною.

- Блокчейн-мережами **консорціуму** керує група організацій. Попередньо відібрані організації поділяють відповідальність за підтримку блокчейну та визначення прав доступу до даних. Галузі, в яких багато організацій мають спільні цілі та отримують вигоду від спільної відповідальності, часто віддають перевагу блокчейн-мережам консорціуму. Наприклад, Global Shipping Business Network Consortium – це некомерційний блокчейн-консорціум, який прагне оцифрувати судноплавну галузь і посилити співпрацю між операторами морської галузі [29].

2.3 Типові застосування технології Blockchain

Хоча блокчейн переважно відомий своїми застосуваннями у фінансовому секторі, наприклад, криптовалютах і цифрових платежах, він також має

потенційні нефінансові застосування, які можуть принести користь різним галузям і секторам. До них можна віднести наступне:

- **Управління ідентифікацією та автентифікація**: як вже було сказано в 2.1 Blockchain може забезпечити безпечний і децентралізований спосіб перевірки та керування цифровими ідентифікаторами, не покладаючись на централізовані треті сторони чи бази даних. Blockchain також може дозволити користувачам контролювати власні дані та конфіденційність, а також надавати або скасовувати доступ до своєї інформації за бажанням.

- **Управління ланцюгом поставок**: блокчейн може підвищити ефективність і відстежуваність ланцюгів поставок, забезпечуючи спільний і незмінний запис про походження, рух і якість товарів і послуг [30].

- **Охорона здоров'я**: блокчейн може покращити безпеку та взаємодію медичних записів, дозволяючи пацієнтам і постачальникам отримувати доступ до своїх даних і обмінюватися ними на різних платформах і системах. Блокчейн також може забезпечити більш точну та своєчасну діагностику, лікування та профілактику захворювань, а також сприяти медичним дослідженням та інноваціям [31, 32].

- **Державні послуги**: блокчейн може підвищити ефективність і прозорість урядових операцій, зменшивши бюрократію, корупцію та людські помилки. Блокчейн також може підвищити участь громадян і довіру, забезпечуючи більш безпечні та перевірені системи голосування, записи власності чи документи, що посвідчують особу [31, 32].

- **Інтернет речей (IoT)**: Blockchain може дозволити пристроям IoT через Інтернет надсилати дані до приватних мереж блокчейну для створення захищених від втручання записів спільних транзакцій. Блокчейн також може забезпечити додаткову безпеку, підзвітність і гнучкість для додатків Інтернету речей у різних областях, таких як розумні будинки, розумні

міста, розумне виробництво, розумне сільське господарство, розумна енергетика, розумна охорона здоров'я, розумний транспорт тощо [33, 34].

Наразі недостатньо досліджень по використанню технології Blockchain саме в SPAN мережах, однак, оскільки SPAN це децентралізована розподілена мережа, прикладом можуть бути існуючі застосування в Інтернеті речей (IoT), оскільки підхід цієї технології подібний в тому сенсі, що він також передбачає ці підходи до побудови мереж, пристроями в яких є розумні прилади, в тому числі смартфони. Оскільки дослідження і пропозиції реалізації технології Blockchain в IoT активно з'являються, пропонується для прикладу проаналізувати використання Blockchain саме в IoT.

2.4 Аналіз використання технології Blockchain для IoT

Існує кілька визначень IoT, серед яких можна виділити визначення від Ernst and Young (EY), яке передає суть технології: «Інтернет речей (IoT) описує підключення пристроїв – будь-яких пристроїв – до Інтернету за допомогою вбудованого програмного забезпечення та датчиків для спілкування, збору та обміну даними один з одним».

IoT означає мережу підключених пристроїв, які здатні збирати та обмінюватися даними. Платформи, що підтримують IoT, надають загальну мережу для пристроїв для передавання своїх даних і спілкування цих пристроїв один з одним, дозволяючи людям використовувати це в своїх інтересах.

Комунікаційні пристрої, або датчики, вбудовані в повсякденні об'єкти, такі як телефони, телевізори, системи внутрішнього клімату, електроприлади, автомобілі, світлофори та промислове обладнання. Ці датчики постійно видають дані про робочий стан підключених пристроїв і дозволяють їм надсилати та отримувати дані один від одного через хмару (Інтернет). Потім платформи IoT аналізують дані, щоб витягувати цінну інформацію та ділитися нею з іншими пристроями, щоб ініціювати певні команди чи дії. Результатом є кращий досвід роботи з людьми, більша автоматизація та підвищення

ефективності. Наприклад, на виробництві всі різні компоненти та машини на заводі можуть бути оснащені датчиками, які постійно передають дані про стан системи назад до мобільних додатків операторів. Тоді потенційні проблеми можна виявити та усунути до того, як станеться поломка, заощаджуючи компанії час і гроші.

Способів використання IoT майже безліч: від підтримки медичних пристроїв пацієнтів у належному робочому стані до боротьби зі швидким вирубуванням лісів у тропічних лісах по всьому світу [35]. Але мережі IoT не ідеальні. Пристрої постійно обмінюються важливою інформацією через Інтернет, що робить їх основною мішенню для хакерів. Тому конфіденційність і безпека є основними проблемами.

Традиційні системи IoT залежать від централізованої архітектури. Інформація надсилається з пристрою в хмару, де дані обробляються за допомогою аналітики, а потім надсилаються назад на пристрої IoT. Смарт-контракти в блокчейн-мережах дозволять пристроям функціонувати безпечно та автономно шляхом створення угод, які виконуються лише після виконання певних вимог. Це не тільки забезпечує більшу автоматизацію, масштабованість і дешевші передачі (не потрібна третя сторона для нагляду за транзакціями), але ці смарт-контракти також можуть запобігти перевизначенню особами, які хочуть використовувати дані для власної вигоди. Інформація передається через децентралізовану криптографічно захищену мережу, що означає, що суттєво ускладнюється можливість скомпрометувати безпеку мережі. Також, з централізованою мережею ризик того, що одна точка збою виведе з ладу всю мережу, цілком суттєвий. Децентралізована блокчейн-мережа зменшує цей ризик за допомогою мільйонів окремих вузлів, які передають дані на одноранговій основі (p2p), щоб забезпечити безперебійну роботу решти мережі IoT [36].

Наразі використання технології Blockchain в IoT ще не суттєво популярне, однак існує вже чимало проектів та рішень саме з таким підходом.

Можна виділити наступні: IOTA, Helium, Xage, Atonomi, IoTeX, SmartAxiom, UBIRCH, Chain of Things, Riddle&Code, Modium.io, Hdac, VeChain, Waltonchain, Moeco, FOAM та Power Ledger.

- **IOTA** – це розподілений реєстр, створений для «Інтернету всього» – мережі для обміну цінностями та даними між людьми та пристроями [37]. IOTA використовує свою мережу, відому як Tangle, як «основу IoT» із технологією розподіленого реєстру для «захищених від втручання даних, безпомилкових мікротранзакцій і низьких вимог до ресурсів». IOTA також підтримує промисловий IoT з автоматизованою торгівлею даними, товарами та послугами. IOTA – це інфраструктура для пристроїв IoT, яким потрібно обробляти великі обсяги мікроданих. Основними функціями розподіленого реєстру Tangle є зв'язок між пристроями, безкоштовні мікроплатежі та квантово стійкі дані [38].

- **Helium** – це децентралізована мережа пристроїв. Компанія використовує блокчейн для підключення малопотужних пристроїв IoT (наприклад, сенсорів, маршрутизаторів і мікрочіпів) до Інтернету. Інфраструктура бездротового Інтернету Helium, заснована на блокчейні, використовує радіотехнологію для покращення Інтернет-з'єднання та суттєвого зменшення енергії, необхідної для роботи «розумних» пристроїв. Helium розробили власний протокол WHIP, який дозволяє за допомогою радіохвиль передавати сигнал Інтернет віддаленим пристроям використовуючи шлюзи та ретранслятори, що підтримують цей протокол.

- **Xage** – це перша платформа безпеки для IoT, захищена блокчейном. Зосереджуючись на промислових застосуваннях для таких галузей, як сільське господарство, енергетика, транспорт і комунальні послуги, блокчейн Xage забезпечує захист пристроїв IoT від несанкціонованого доступу та доступ до захищених ліній зв'язку між розумними пристроями.

- **Atonomi** використовує технологію Blockchain для забезпечення безпеки пристроїв, підключених до Інтернету речей. Рішення компанії включає перевірку ідентифікації пристрою в незмінному реєстрі, відстеження репутації для виявлення аномалій пристрою та можливість здійснення транзакцій між перевіреними пристроями незалежно від їх виробника. Його можна використовувати для розумних міст, промислових застосувань, охорони здоров'я, розумних будинків і автономних транспортних засобів.

- **IoTeX** розробив платформу на основі блокчейну, яка з'єднує дані та розумні пристрої з децентралізованими додатками блокчейну. Продукт компанії Ucam – це «перша в світі блокчейн-камера домашньої безпеки», яка поєднує Blockchain та IoT, щоб гарантувати користувачам безпечний доступ до відеоматеріалів, знятих у їхніх будинках та навколо них. Існує також Pebble, яка «є безпечною платформою прототипування стільникового Інтернету речей, що працює від акумулятора, розроблена для додатків на основі блокчейну» для відстеження мобільних активів, автоматизації процесів тощо.

- **SmartAxiom** використовує технологію Blockchain для захисту пристроїв IoT. Технологія компанії забезпечує безключову автентифікацію на основі однорангових пристроїв, а також захист від різних кібератак і несанкціонованого доступу до пристроїв.

- Платформу довіри **UBIRCH** можна інтегрувати з платформами Інтернету речей, щоб створити те, що компанія називає «ланцюгом довіри», що означає, що під час збору даних IoT використовуються технології Blockchain та криптографії, щоб гарантувати, що дані не можуть бути змінені [39].

- **Chain of Things (CoT)** – це консорціум технологів і блокчейн-компаній, що досліджує найкращі варіанти використання поєднання Blockchain та

Інтернету речей, що можуть запропонувати значні переваги промисловим, екологічним і гуманітарним застосуванням. Наразі CoT створила Maru – інтегроване апаратне забезпечення для блокчейну та Інтернету речей для вирішення проблем із ідентифікацією, безпекою та сумісністю. Існує три розроблених варіанти використання, які називаються Chain of Security, Chain of Solar і Chain of Shipping [40].

- **Riddle&Code** надає рішення для криптографічного тегування для блокчейнів у інтелектуальній логістиці та управлінні ланцюгами поставок. Працюючи над інтеграцією між пристроями IoT і мережами розподіленого реєстру, Riddle&Code пропонує комбіноване запатентоване апаратне та програмне рішення, яке забезпечує безпечну та надійну взаємодію з пристроями IoT, надаючи пристроям «надійну цифрову ідентифікацію».
- **Modum.io** поєднує датчики IoT із технологією Blockchain, забезпечуючи цілісність даних для транзакцій із фізичними продуктами. Датчики Modum реєструють умови навколишнього середовища, такі як температура, яким товари піддаються під час транспортування. Коли товар надходить до наступної транзитної точки або кінцевого клієнта, дані датчика перевіряються на відповідність заздалегідь визначеним умовам у смарт-контракті на блокчейні. Контракт підтверджує, що умови відповідають усім вимогам, встановленим відправником, його клієнтами або регулятором, і запускає різні дії, такі як сповіщення відправника та одержувача, оплата або випуск товарів [38].
- Компанія **Hyundai Digital Asset Company (Hdac)** застосовує технологію Blockchain для швидкого й ефективного зв’язку, перевірки особи, автентифікації та зберігання даних між пристроями IoT. Система включає гібридну мережу (публічну та приватну) для збільшення швидкості та обсягу транзакцій, що робить її корисною для пристроїв IoT. Технологія

застосовується до розумних фабрик, розумних будинків і розумних будівель для міжмашинних транзакцій і роботи між пристроями IoT.

- **VeChain** – це глобальна публічна блокчейн-платформа корпоративного рівня. Блокчейн використовується різними способами, особлива увага придається розширеній інтеграції Інтернету речей у логістиці холодового ланцюга за допомогою використання фірмових пристроїв Інтернету речей для відстеження ключових показників, таких як температура, протягом усього шляху. Крім того, платформа може зберігати автомобільні документи, створюючи цифрові записи автомобілів – включаючи історію ремонту, страхування, реєстрацію та навіть поведінку водія протягом усього життєвого циклу. VeChain також використовує технологію IoT для предметів розкоші, вбудовуючи розумні чіпи в продукти розкоші, щоб бренди могли контролювати свої канали продажів у режимі реального часу, тим самим запобігаючи незаконній торгівлі товарами та надаючи споживачам можливість перевірити справжність продукту розкоші.

- **Waltonchain** створено за допомогою поєднання технологій RFID і Blockchain для ефективної інтеграції IoT. Вони в основному зосереджені на відстеженні процесів і продуктів у ланцюжку постачання, де технологія може бути застосована для високоякісної ідентифікації одягу, відстеження харчових продуктів і ліків і відстеження логістики шляхом імплантації RFID-міток і чіпів керування зчитувачем-записувачем у продукти. Потім інформація про статус продуктів завантажується для аналізу в блокчейн [36].

- Ідея **Moeco** полягає в радикальному зниженні вартості інфраструктури IoT. Вони допомагають компаніям оновлювати та підключати свою інфраструктуру за допомогою поєднання неінвазивних датчиків і автоматизованого збору даних із практичними статистичними даними. Варіанти використання клієнтами включають деякі з наступного: запобігання крадіжкам, геозонування розташування та умови зберігання

та транспортування, такі як температура, вологість, світло та виявлення ударів [41, 42].

- **FOAM** надає інструменти для створення краудсорсингової карти та децентралізованих служб визначення місцезнаходження. FOAM вирішує проблему перевірки місцезнаходження за допомогою альтернативної, стійкої до збоїв системи, що надається через відкриту мережу наземних радіостанцій, якою може керувати кожен. FOAM побудовано на блокчейні Ethereum. Токен FOAM дає учасникам контроль над своєю мережею розташування та вимагає від них прийняття рішень, які впливають на її майбутнє. Використання в Інтернеті речей полягає у перевірці безпечного місцезнаходження та локалізації для мереж пристроїв IoT [43].
- **Power Ledger** розробляє програмні рішення на основі Blockchain для відстеження та торгівлі відновлюваною енергією [44]. Як приклад, у 2018 році австралійська фірма Vicinity заключила договір з Power Ledger, що дозволить Vicinity керувати розподілом енергії в режимі реального часу, вирішуючи, чи продовжувати використовувати сонячні батареї, чи перейти до національної електромережі [45].

Відповідно, враховуючи все сказане, побудуємо Таблицю 2, в яку внесемо цільові застосування технології Blockchain, які використовує той чи інший проект. Ті застосування, які використовуються більш чим одним проектом виділяються, тобто безпека, економічна ефективність, відстеження транзакцій, розподіл енергії, решта вказується навпроти відповідного проекту в комірці стовпця Інше. Якщо проект використовує одне із застосувань, у відповідній комірці ставиться «+». Вкінці пропонується підрахувати кількість «+» для цільових застосувань технології Blockchain та обрати основні.

Як видно з Таблиці 2 та Рисунку 11, дванадцять з шістнадцяти рішень описують як одне з цільових застосувань безпеку, три відстеження транзакцій,

по два на економічну ефективність та розподіл енергії та решта мали інше застосування. Можна зробити висновок, що перевагою використання технології Blockchain є перш за все підвищення безпеки систем.

Враховуючи це, та інші особливості та переваги використання технології Blockchain, описані в цьому розділі, для вирішення піднятої в 1 розділі проблеми, пропонується для автентифікації використати технологію Blockchain.

Таблиця 2 – Порівняння використання технології Blockchain в IoT

	Цільове застосування				
Назва	**Безпека**	**Економічна ефективність**	**Відстеження транзакцій**	**Розподіл енергії**	**Інше (вказано)**
Helium	+				організація зв'язку з мережею Internet
Xage	+				
Atonomi	+				
IOTA	+	+			
IoTeX	+				
SmartAxiom	+				
UBIRCH	+				
Modium.io	+				
Riddle&Code	+				
Hdac	+				зберігання
Chain of Things (CoT)	+			+	покращення сумісності
Power Ledger			+	+	
VeChain			+		
Waltonchain			+		
Moeco		+			
FOAM	+				безпечне визначення місцезнаходження

Для більшої наочності побудуємо гістограму даних з Таблиці 2:

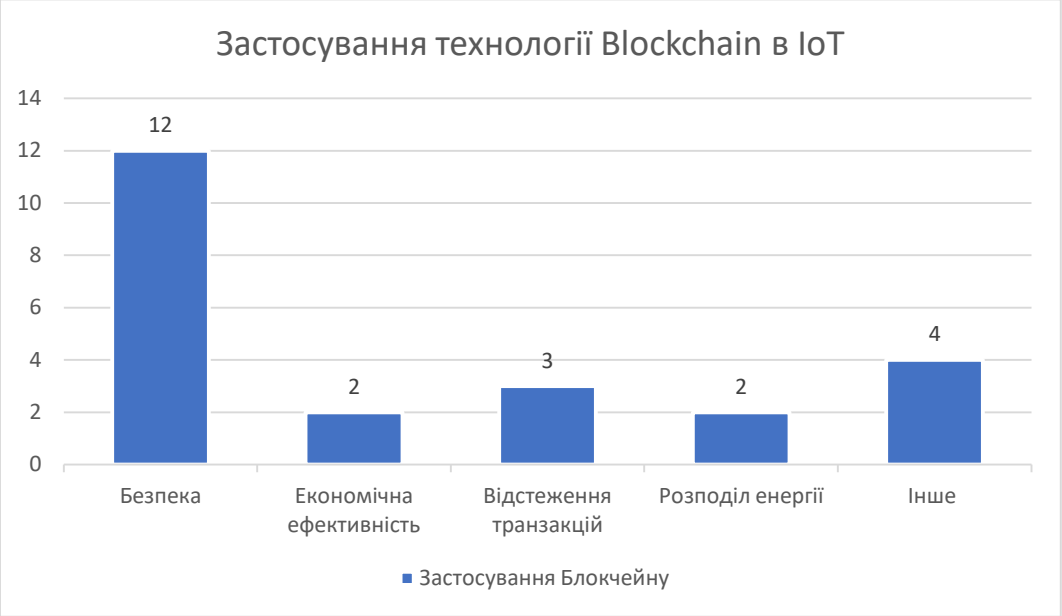

Рисунок 11 – Результати порівняння застосувань Blockchain в IoT

Розділ 3. Автентифікація за допомогою технології Blockchain

Для прикладу використання технології Blockchain для автентифікації в роботі розроблено Android застосунок, що симулює, яким чином може відбуватись автентифікація за допомогою технології Blockchain в Android застосунках, у тому числі і тих, що реалізують SPAN мережі. Для максимального покриття усіх платформ в даному випадку цей застосунок складається з двох частин: сам застосунок на Android, та серверна складова – веб-додаток, що виконує основну логіку автентифікації за допомогою Blockchain.

Для виконання процесу автентифікації за допомогою блокчейну необхідними є дві складові: клієнтська (засіб, що дозволяє виконати автентифікацію з боку користувача) та програмна (засоби, що дозволяють виконати автентифікацію з боку застосунку, що працює на блокчейні).

Для виконання клієнтської частини процесу автентифікації за допомогою Blockchain можуть використовуватися різні засоби. Одним з поширених таких засобів для мобільних застосунків, який пропонується використати в роботі є MetaMask. Цей засіб є Web3 гаманцем, доступним як розширення до браузерів або як застосунок на Android чи iOS, який також є свого роду шлюзом до

застосунків, що працюють на блокчейні, тобто він дозволяє підключатись та взаємодіяти з такими застосунками. За замовчуванням цей засіб використовує мережу Etherium, однак за бажанням користувач може під'єднатися до будь-якої з підтримуваних мереж блокчейну (Polygon, Ropsten, Kavan, Rinkeby, Goerli, Avalance, Arbitrum, BNB, Fantom Opera, Harmony, Optimism, Palm) або ж до власної мережі, для чого потрібно вказати назву, URL та Chain ID [46].

При створенні облікового запису в MetaMask, створюється унікальна публічна адреса, що є шістнадцятковим значенням з 32 символів (Рисунок 12).

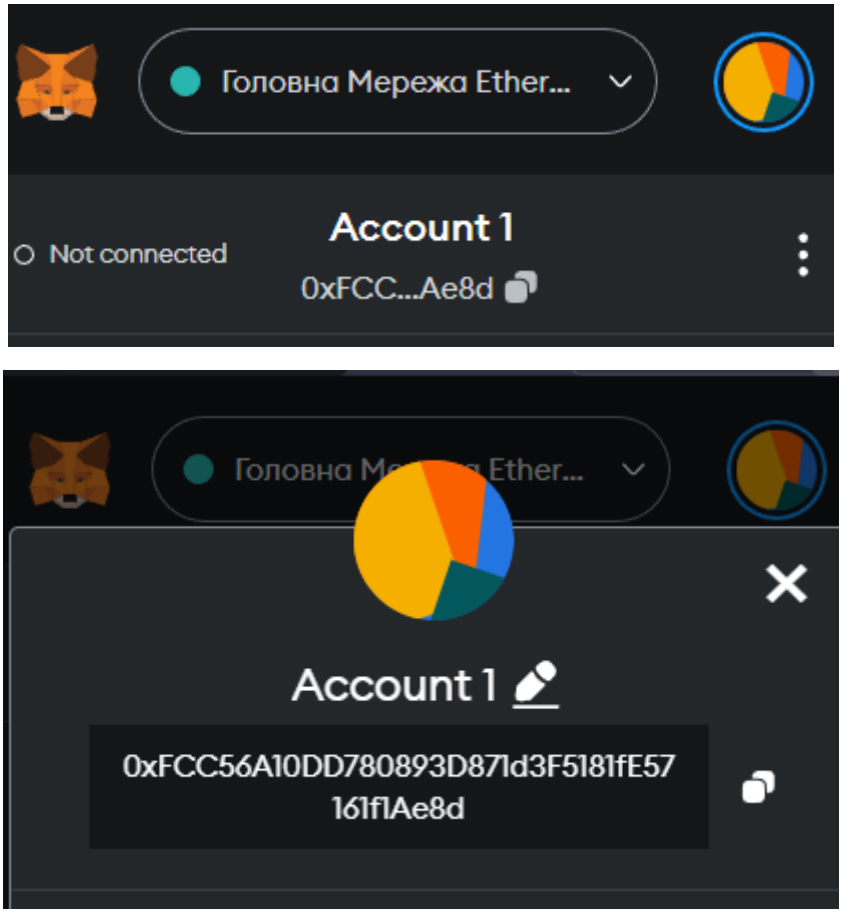

Рисунок 12 Створений обліковий запис в MetaMask

Крім того, для першого створення облікового запису, окрім паролю, заданого користувачем при реєстрації, застосунком генерується так звана секретна фраза для відновлення, що складається зі впорядкованих 12 слів (Рисунок 13), які потрібно запам'ятати або зберегти в безпечному місці і лише вона може використовуватись для відновлення облікового запису в MetaMask.

Також при кожному вході в застосунок необхідне використання паролю або ж біометрії, якщо її підтримує пристрій. Один обліковий запис та адреса можуть бути під'єднані до будь-якої кількості мереж водночас. Цей засіб також надає можливість автентифікуватись до застосунків, що використовують блокчейн за допомогою протоколу WalletConnect, який також є відкритим стандартом для підключення Web3 гаманців до застосунків, що працюють на блокчейні [47].

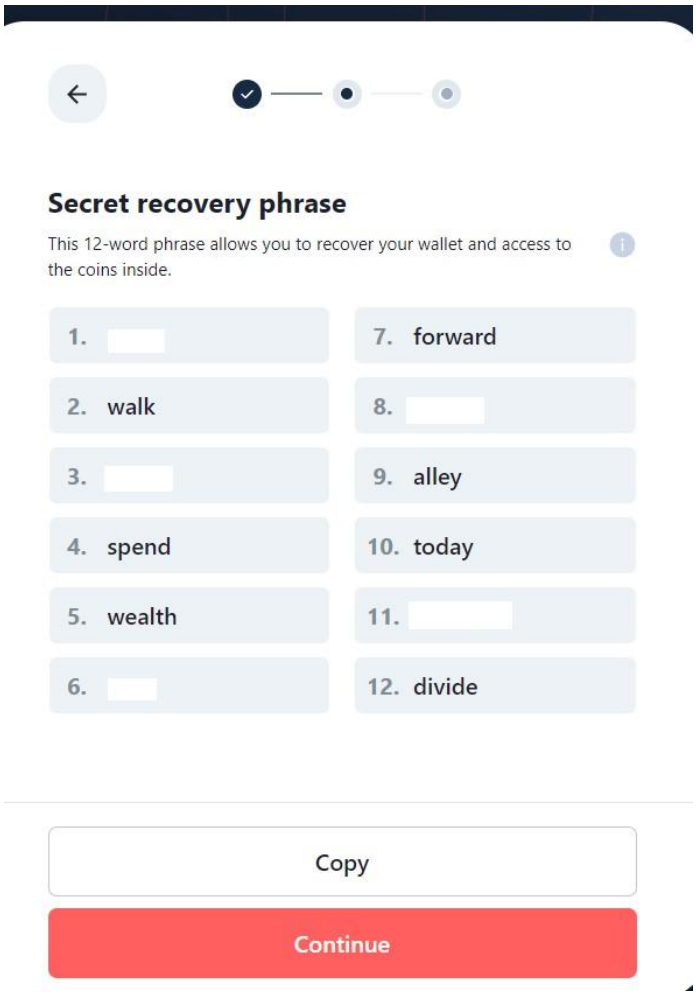

Рисунок 13 Секретна фраза для відновлення облікового запису в MetaMask

Також для виконання процедури автентифікації необхідним є механізм встановлення зв'язку та надсилання повідомлень, необхідних для автентифікації, між застосунком, що працює на блокчейні та автентифікатором (в даному випадку MetaMask). Такий механізм може надати API від Moralis, який дозволяє користувачеві автентифікувати та перевіряти підписані повідомлення за допомогою своїх Web3 гаманців під час використання

програми. Цей API створений відповідно до стандарту EIP-4361 [48] та працює з обраним гаманцем MetaMask. Автентифікація за допомогою цього стандарту працює наступним чином: гаманець надає користувачеві структуроване текстове повідомлення або еквівалентний інтерфейс для підпису з форматом підписаних даних ERC-191. Повідомлення повинне включати адресу, домен, який запитує підпис, версію повідомлення, ідентифікатор мережі Chain ID, uri, nonce (рандомізований токен, що зазвичай обирає перевіряюча сторона та використовується для запобігання атакам повтору, який складається з щонайменше 8 буквено-цифрових символів), і мітку часу видачі. Потім підпис надається перевіряючій стороні, яка перевіряє дійсність підпису та вміст повідомлення (в даному випадку це Moralis) [49].

Приклад запиту на автентифікацію за допомогою такого згенерованого такого повідомлення наведено на рисунку 14. В даному випадку домен вказано згори, він має співпадати зі вказаним URI, MetaMask вказує, що адреса під питанням це Account 1, що відповідає вказаному на рисунку 12, вказано версію повідомлення (Version: 1), ідентифікатор мережі (Chain ID: 137), згенерований nonce, що в даному випадку перевищує 8 символів для підвищення безпеки і також видно мітку часу видачі (Issued At).

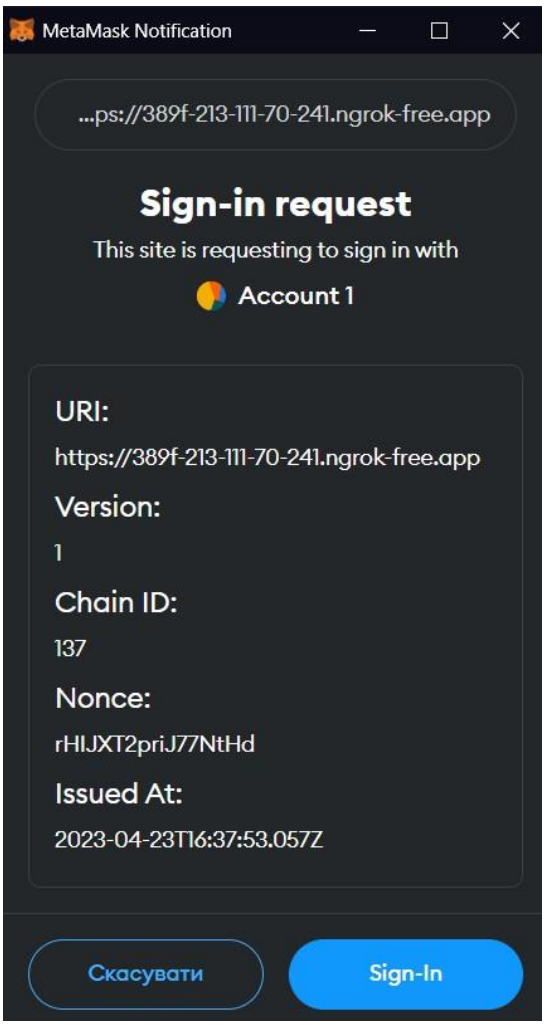

Рисунок 14 Запит на автентифікацію відповідно до стандарту EIP-4361

Крім цього, MetaMask на Android має додаткову вимогу до застосунків, яка не присутня у версії MetaMask для браузерів на ПК, в тому сенсі, що розподілений застосунок, що працює на блокчейні, до якого необхідно під'єднатися та автентифікуватися повинен мати SSL захист, тобто використовувати HTTPS, а не HTTP. Також для перевірки на пристрої Android, застосунок повинен мати домен, тобто якщо запустити застосунок локально на ПК на http://localhost:3000, MetaMask такий додаток якщо і знайде, не зможе відкрити. Одним можливим рішенням є зробити самопідписаний сертифікат, однак все ще буде присутня необхідність існування публічного домену. В рамках даної роботи пропонується використати Ngrok, що дозволяє розробникам транслювати застосунки, запущені на localhost та заданому порту в безкоштовний згенерований публічний домен, до якого є доступ зі всього

світу. Єдиною проблемою тут є те, що після кожного запуску такої трансляції домен регенерується, тому кожен раз необхідно оновлювати новий домен в коді програми для роботи застосунку. Для роботи Ngrok необхідно мати обліковий запис на ngrok.com, у випадку ОС Windows необхідно завантажити файл ngrok.exe та виконати команду

./ngrok.exe config add-authtoken <токен>,

Сам токен можна отримати на веб-сайті після реєстрації облікового запису. Після цих дій трансляція виконується виконанням команди

./ngrok.exe http <порт>,

Результат виконання цієї команди на порт 3000 наведено на рисунку 15

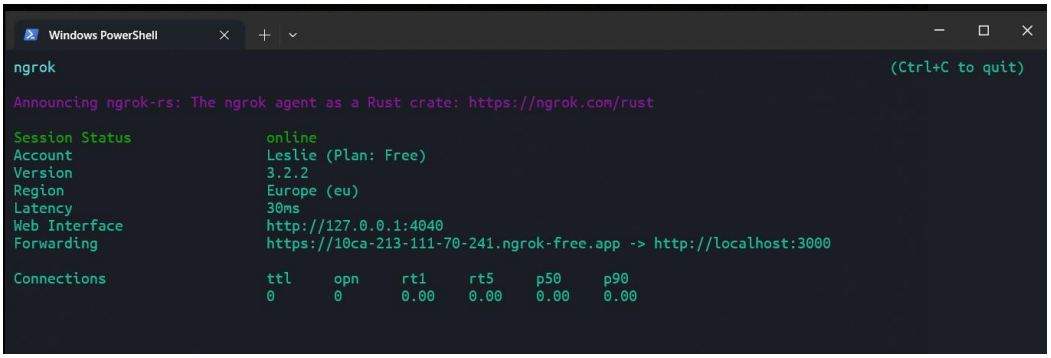

Рисунок 15 Запуск Ngrok для трансляції localhost в згенерований публічний домен з https

Відповідно, для реалізації описаної процедури автентифікації пропонується розробити застосунок, що використовує MetaMask та Moralis, який складається з двох складових: перша складова це додаток, написаний на мові програмування Java, у якості середовища розробки та виконання пропонується використати Android Studio, як основне середовище для програмування застосунків на Android – ця складова відіграє роль програмної реалізації SPAN, до якої буде задіяна автентифікація; друга частина це веб-застосунок, написаний на NextJS, у якості середовища розробки та виконання пропонується використати Visual Studio Code, що є безкоштовним середовищем яке показує дерево програми, показує синтаксис багатьох мов програмування та

дощволяє використовувати консоль всередині середовища – ця складова використовує Ngrok для отримання публічного захищеного домену, до неї під’єднується перша складова і на ній відбувається взаємодія з MetaMask для автентифікації за допомогою API від Moralis.

Для початку для розуміння роботи програми пропонується розглянути блок-схему наведену на рисунку 16, що описує принцип роботи отриманої реалізації описаної процедури автентифікації. Принцип роботи наступний: коли користувач відкриває застосунок вперше йому пропонується автентифікуватися за допомогою MetaMask. Коли користувач погоджується, відкривається веб-додаток всередині MetaMask. На початку MetaMask відправляє адресу користувача (Address) та інформацію про мережу (Chain ID) в додаток, після чого за допомогою Moralis API застосунок викликає Moralis за адресою, вказаною на блок-схемі, куди передаються такі дані як отримані на попередньому кроці Address, Chain ID, а також домен та URL самого застосунку та мітку часу (Timeout). Після цього, Moralis генерує повідомлення, яке користувач повинен підписати. Для цього повідомлення генеруєтсья новий профіль (Profile ID), який потім буде прив’язано до даного користувача. Після цього повідомлення для підписання відправляється до застосунку, який потім його передає до клієнта, тобто, в даному випадку MetaMask, щоб користувач його підписав. Після цього, якщо користувач не підписав повідомлення, йому повідомляється про необхідність виконати процедуру знову, якщо ж повідомлення підписане, саме повідомлення та підпис надсилається застосунком за допомогою API до Moralis за посиланням на блок-схемі. Далі Moralis виконує верифікацію підпису в раховуючи створений раніше параметр Timeout і якщо він не дійсний, процедуру потрібно виконати знову якщо ж дійсний, застосунок додає користувача та генерує зашифрований JWT токен, що зберігається в створеному файлі cookie. Цей токен також можна зберігати в базі даних але в рамках даного прикладу пропонується використати саме такий підхід який не вимагає додаткового створення бази даних. Після успішної

автентифікації користувачу показується дані про нього, такі як адреса, ProfileID, які можна верифікувати за допомогою Moralis та підпис.

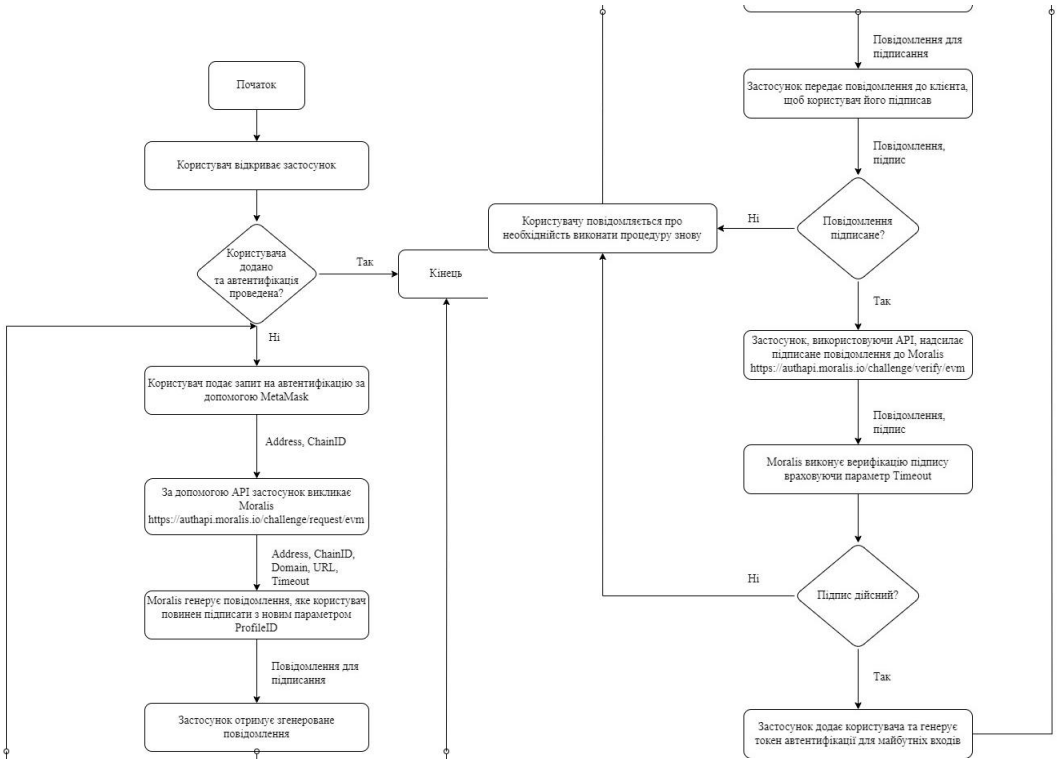

Рисунок 16 Блок-схема реалізації автентифікації за допомогою технології Blockchain

Тепер пропонується розглянути сам процес автентифікації з точки зору користувача в запущеному додатку, відкритому на пристрої Android 11. Коли користувач відкриває застосунок в нього є дві опції: виконати процес автентифікації (Sign Up), якщо така ще не виконана або увійти в додаток якщо вона вже була пройдена (Sign In), для чого необхідно ввести Profile ID та пов’язану адресу користувача, які можна отримати пройшовши процес автентифікації. Вигляд відкритого застосунку наведено на рисунку 17. Як можна бачити, користувача інформовано про кроки, які потрібно виконати, тобто ввести потрібні дані та натиснути кнопку «Sign In» або виконати автентифікацію натиснувши «Sign Up!».

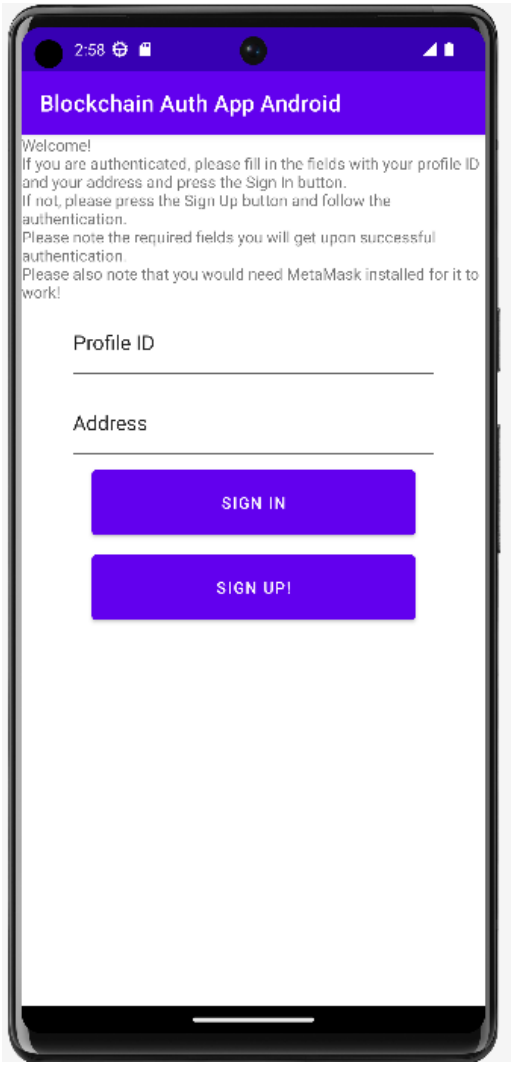

Рисунок 17 Вигляд Android застосунку при відкритті

При натисненні кнопки «Sign Up» користувач відкриває веб-додаток в MetaMask, він бачить сторінку наведену на рисунку 18а.

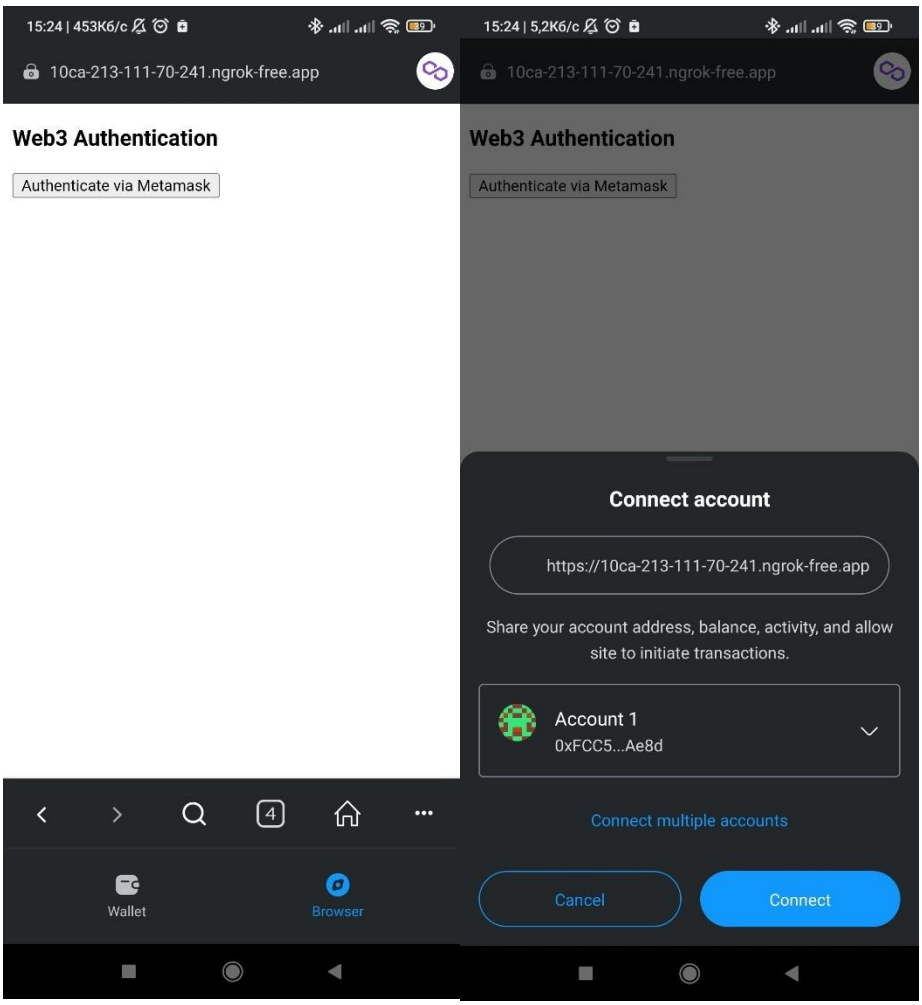

Рисунок 18 (а) Сторінка автентифікації у веб-додатку - зліва,
(б) Натискання кнопки «Authenticate via Metamask» - справа.

Коли користувач натискає кнопку «Authenticate via Metamask» йому пропонується під’єднати обліковий запис до застосунку. На цьому етапі можна обрати обліковий запис або під’єднати декілька водночас, як видно на рисунку 18б. У випадку згоди обліковий запис підключається до додатку та наступним кроком додаток надсилає користувачу повідомлення для підписування (рисунок 19а), яке можна більш детально подивитись (рисунок 19б). Як можна бачити на рисунку 19б, всі дані, які вимагаються стандартом EIP-4361 присутні.

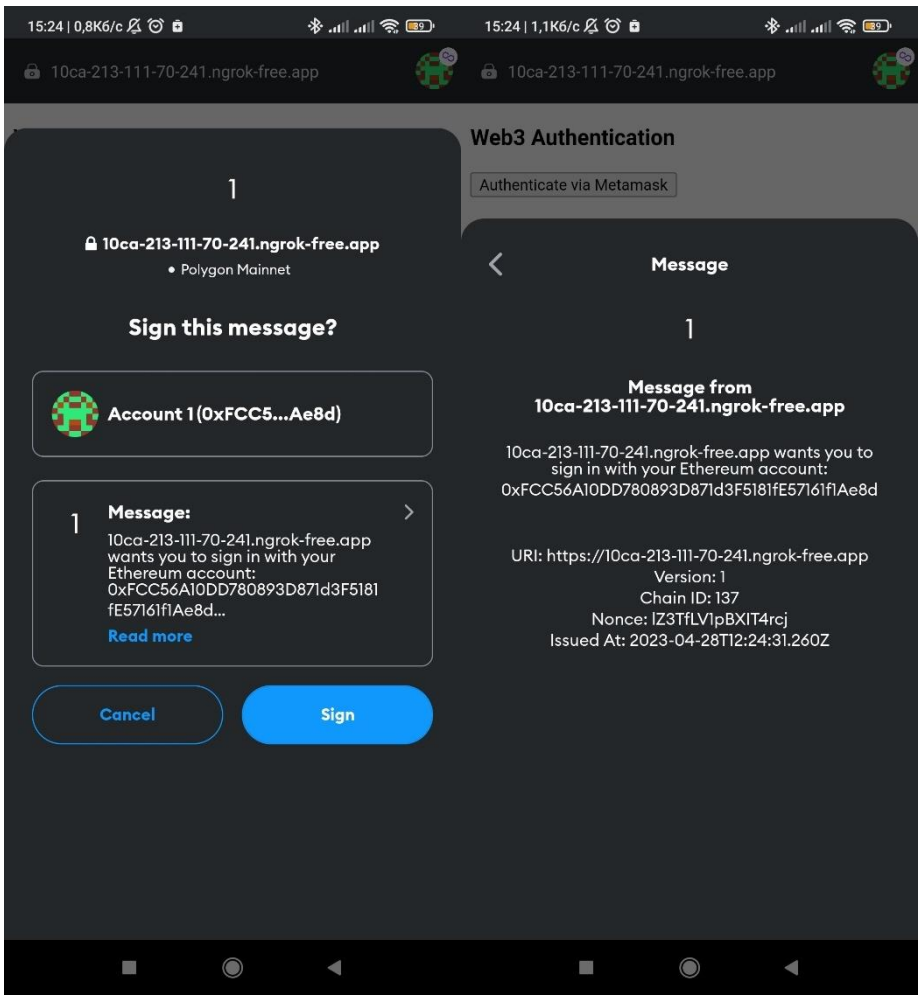

Рисунок 19 (а) Повідомлення для підписування,
(б) розгорнуте повідомлення в деталях

Після підписання повідомлення користувача успішно автентифіковано і застосунок його перенаправляє на сторінку /user, яка показує дані про користувача (рисунок 20). Тут можна бачити ті ж самі дані, що були в повідомленні для підписання, а також згенерований profileId, який асоціюється із заданою адресою користувача (поле address).

У самому застосунку MetaMask користувач потім може переконатися, що його обліковий запис підключено до додатку (рисунок 21а) та за бажання відкликати наданий підпис (рисунок 21б), після чого необхідно буде знову прийти процес автентифікації.

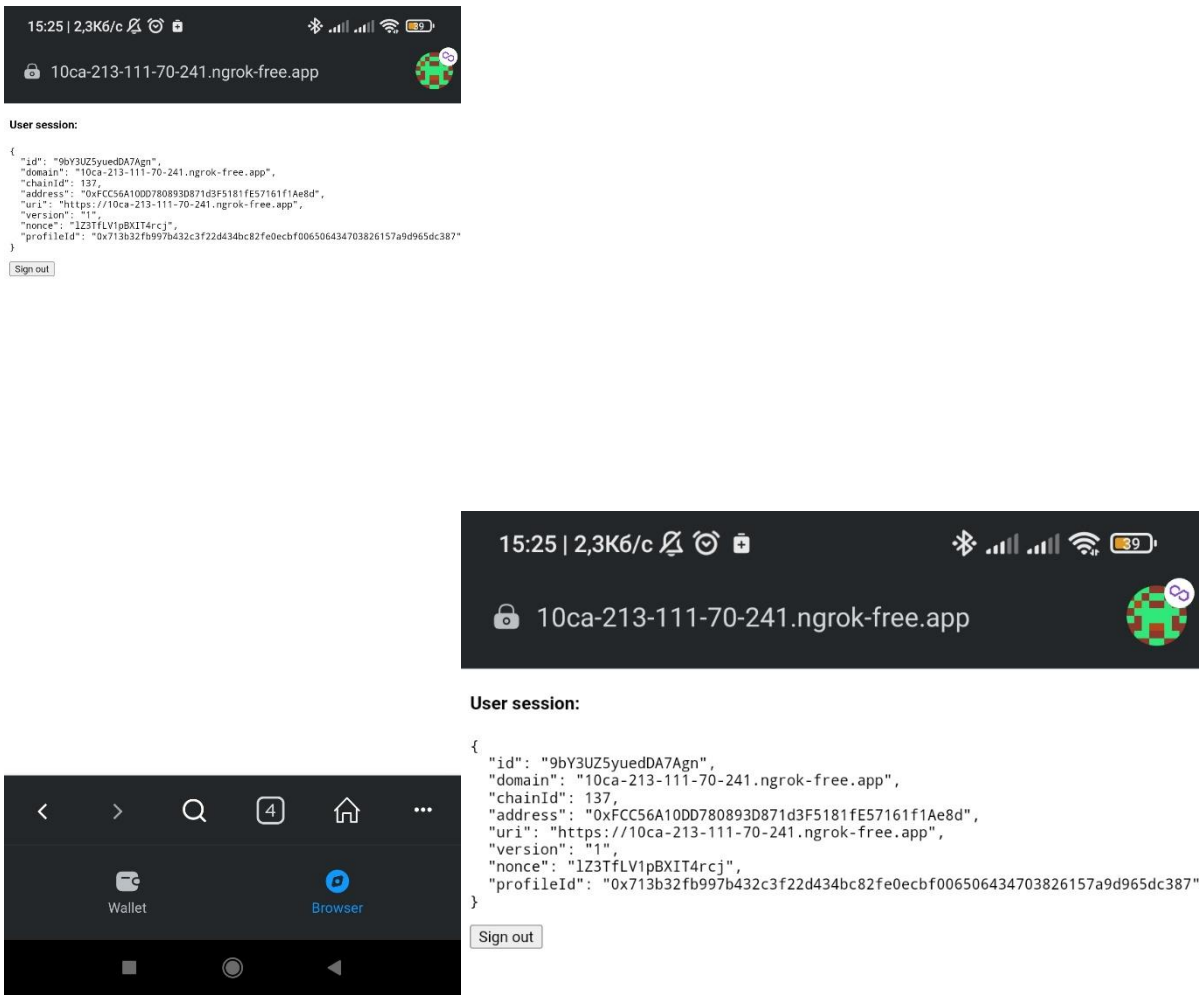

Рисунок 20 Сторінка автентифікованого користувача

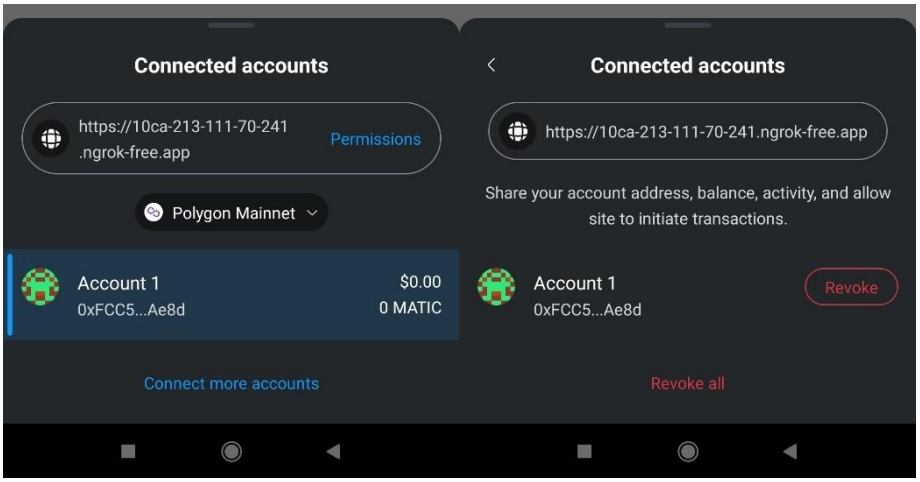

Рисунок 21 (а) Перевірка, що додаток під'єднано

(б) Опція відкликання підпису про надання дозволів

Тепер, коли в користувача є Profile ID, він може повернутись до Android застосунку і ввести туди цей параметр та адресу, як показано на рисунку 22а.

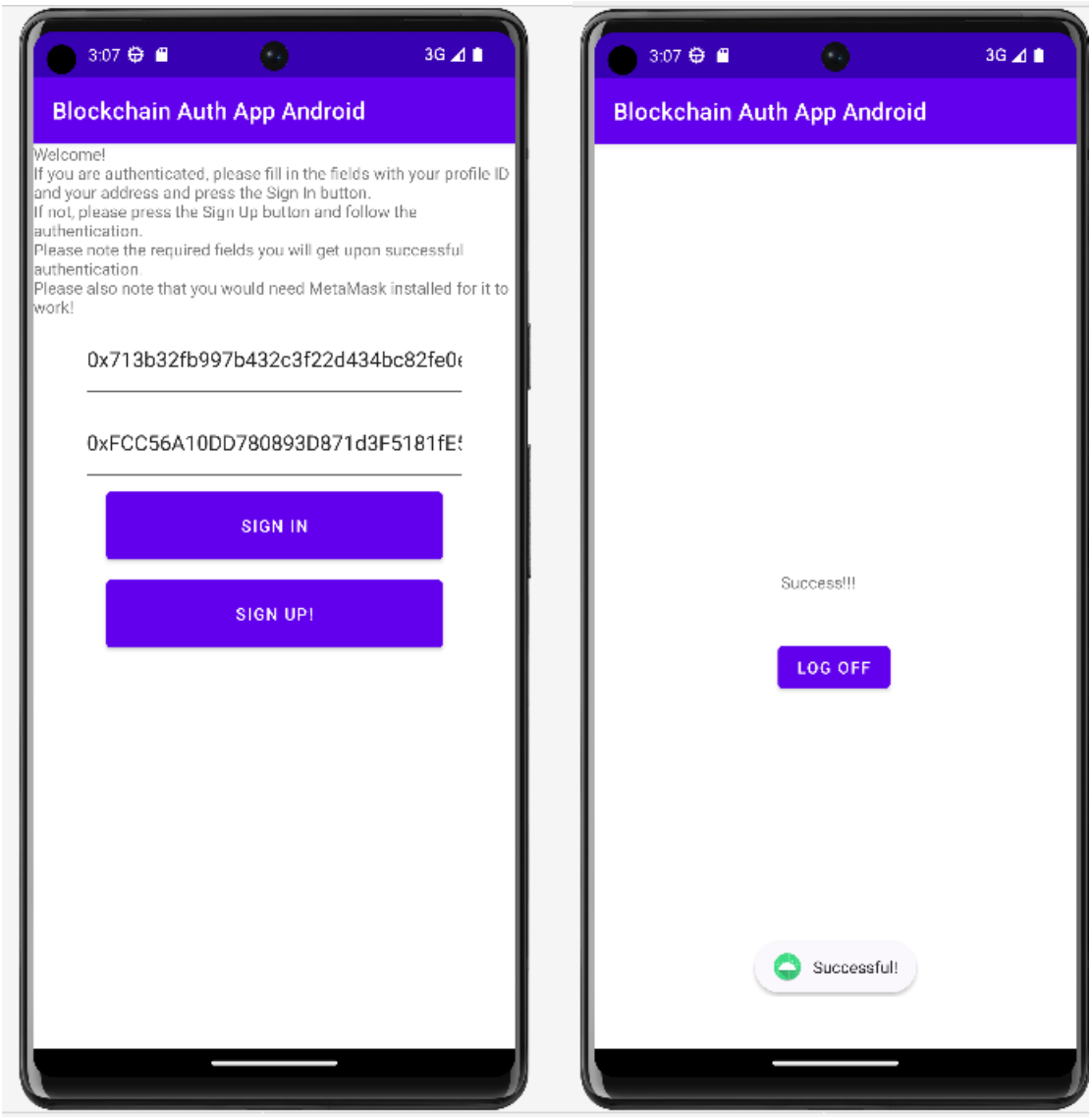

Рисунок 22 (а) внесення необхідних параметрів

(б) успішний вхід в застосунок

За успішної перевірки користувач входить в застосунок і, за бажання може вийти з нього при натисненні кнопки «Log Off», як показано на рисунку 22б.

Сам код застосунку наведено на GitHub в [50]. Однак для наочності наведемо основні фрагменти, що забезпечують логіку роботи рішення. Натискання кнопки «Sign In» виконується фрагментом коду, вказаному в додатку А. Натискання кнопки «Sign Up» відкриває посилання з веб-додатком. Саме посилання враховуючи описаний раніше недолік Ngrok потрібно оновлювати після кожного перезапуску Ngrok трансляції. Якщо говорити про

роботу веб-застосунку, варто вказати такий важливий файл, як .env.local, який містить дані про ключ Api (MORALIS_API_KEY) для роботи Moralis, секретне слово (NEXTAUTH_SECRET), що використовується для шифрування токенів JWT користувачів, він може бути будь-яким однак в даному випадгу згенерований за допомогою openssl rand -hex 32, ім'я домену (APP_DOMAIN) та посилання, за яким працюватиме додаток (NEXTAUTH_URL), які подібно до «Sign Up» потрібно змінювати при перезапуску Ngrok. Цей фрагмент на момент написання наведено в додатку Б. Цей файл використовується для уникнення поширення критичних даних в код та зазвичай не публікується в GitHub. Структура веб-додатку це дві сторінки: головна сторінка на якій виконується автентифікація та описується файлом index.tsx, та сторінка з даними про автентифікованого користувача /user, а саме user.jsx. Основна частина, тобто процедура автентифікації, описана на блок-схемі на рисунку 16 відбувається саме на головній сторінці. Код файлу index.tsx наведено в додатку В. Коли користувача успішно автентифіковано, він переходить на сторінку /user, де отримує profileId. Код файлу user.jsx наведено в додатку Г. Також робота API від Moralis передбачає використання ще двох файлів, а саме […nextauth].ts, який забезпечує роботу з JWT токенами та […moralis].ts, що використовується для формування повідомлення для підписування та використовує параметри з файлу .env.local. Ці файли наведені в додатках Ґ та Д відповідно.

ВИСНОВКИ

В даній роботі розглянуто та проаналізовано застосунки, що є програмними реалізаціями такої технології як спеціальні мережі смартфонів, або SPAN, які здатні працювати як за наявності мережі Інтернет, так і без неї, що стали актуальними в ситуаціях з тимчасовими вимкненнями електроенергії з тієї чи іншої причини.

В роботі визначено, що такі застосунки мають проблему з ненадійною процедурою автентифікації користувачів: для створення нового користувача найбільш захищеним методом, що використовується, є використання паролю, при втраті якого доступ до облікового запису користувача відновити неможливо.

Відповідно, в роботі запропоновано рішення з використанням технології Blockchain для автентифікації в таких застосунках. Таке рішення використовує для автентифікації захищений Web3 гаманець MetaMask та API від Moralis, за допомогою якого відбувається автентифікація та верифікація користувача на блокчейні. Таке рішення направлене на підвищення безпеки користувачів таких застосунків і пропонується до використання за наявності мережі Інтернет.

В роботі показано рішення, яке передбачає використання публічної мережі блокчейн, на кшталт Etherium чи Polygon, що може потребувати вдосконалення шляхом побудови окремої персональної мережі на блокчейні, що працюватиме виключно для потреб рішення. Також, в роботі передбачене збереження токенів про користувача у файлах cookie, це збереження можна робити у спеціально виділеній базі даних, яка працюватиме на такій персональній мережі блокчейн.

У рамках виконання дипломної роботи магістра прийнято участь у двох міжнародних конференціях, що відбуваються на факультеті радіофізики, електроніки та комп’ютерних систем [51, 52].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Toh, Chai Keong, Ad Hoc Mobile Wireless Networks // Prentice Hall Publishers 2002
2. R. Agrawal, N. Faujdar, Classification and comparison of ad hoc networks: A review // Egyptian Informatics Journal, 2023, No. 1 Vol. 24. P. 1-25.
3. Macrotrends, World Population 1950-2023 [Електронний ресурс] - Режим доступу до ресурсу: https://www.macrotrends.net/countries/WLD/world/population (Рік звернення 2023)
4. Statista, Number of smartphone subscriptions worldwide from 2016 to 2021, with forecasts from 2022 to 2027 [Електронний ресурс] - Режим доступу до ресурсу:https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide (Рік звернення 2023)
5. Ericsson, Ericsson Mobility Visualizer [Електронний ресурс] - Режим доступу до ресурсу: https://www.ericsson.com/en/reports-and-papers/mobility-report/mobility-visualizer (Рік звернення 2023)
6. Statista, Forecast of smartphone user numbers in Ukraine from 2018 to 2027 [Електронний ресурс] - Режим доступу до ресурсу:https://www.statista.com/statistics/1134645/predicted-number-of-smartphone-users-in-ukraine/ (Рік звернення 2023)
7. Державна служба статистики України, Чисельність населення по регіонах (за оцінкою) на 1 лютого 2022 року та середня чисельність у січні 2022 року [Електронний ресурс] - Режим доступу до ресурсу:http://db.ukrcensus.gov.ua/PXWEB2007/ukr/news/op_popul.asp (Рік звернення 2023)

8. Веб-Архів, FireChat - Open Garden [Електронний ресурс] - Режим доступу до ресурсу:https://web.archive.org/web/20190228093501/https://www.opengarden.com/firechat/ (Рік звернення 2023)
9. Веб-Архів, 404 Not Found - Open Garden [Електронний ресурс] - Режим доступу до ресурсу:https://web.archive.org/web/20200328142351/https://opengarden.com/firechat (Рік звернення 2023)
10. Open Garden, 403 Forbidden [Електронний ресурс] - Режим доступу до ресурсу:https://opengarden.com/ (Рік звернення 2023)
11. Веб-Архів, FireChat on the App Store [Електронний ресурс] - Режим доступу до ресурсу:https://web.archive.org/web/20190823100245/https://apps.apple.com/us/app/firechat/id719829352 (Рік звернення 2023)
12. Веб-Архів, FireChat - Android Apps on Google Play [Електронний ресурс] - Режим доступу до ресурсу:https://web.archive.org/web/20161018012250/https://play.google.com/store/apps/details?id=com.opengarden.firechat&referrer=utm_source%3Dopengarden.com%26utm_medium%3Dweb%26utm_campaign%3Dbutton (Рік звернення 2023)
13. Bridgefy, Bridgefy - Offline Messages App & SDK [Електронний ресурс] - Режим доступу до ресурсу:https://bridgefy.me/ (Рік звернення 2023)
14. Google, Bridgefy - Offline Messages - Додатки в Google Play [Електронний ресурс] - Режим доступу до ресурсу:https://play.google.com/store/apps/details?id=me.bridgefy.main&utm_medium=website&utm_term=demo-app&utm_content=google-play (Рік звернення 2023)
15. The Serval Project [Електронний ресурс] - Режим доступу до ресурсу:https://www.servalproject.org/ (Рік звернення 2023)

16. Документація Serval Project [Електронний ресурс] - Режим доступу до ресурсу:https://developer.servalproject.org/ (Рік звернення 2023)

17. GitHub, GitHub - servalproject/serval_chat [Електронний ресурс] - Режим доступу до ресурсу:https://github.com/servalproject/serval_chat (Рік звернення 2023)

18. GitHub, GitHub - servalproject/batphone [Електронний ресурс] - Режим доступу до ресурсу:https://github.com/servalproject/batphone (Рік звернення 2023)

19. Веб-Архів, The Serval Mesh - Apps on Google Play [Електронний ресурс] - Режим доступу до ресурсу:https://web.archive.org/web/20181002015939/https://play.google.com/store/apps/details?id=org.servalproject (Рік звернення 2023)

20. GitHub, GitHub - servalproject/serval-dna [Електронний регіонах] - Режим доступу до ресурсу:https://github.com/servalproject/serval-dna (Рік звернення 2023)

21. GitHub, GitHub - servalproject/Serval-Chat-iOS [Електронний регіонах] - Режим доступу до ресурсу:https://github.com/servalproject/Serval-Chat-iOS (Рік звернення 2023)

22. Briar, How it works - Briar [Електронний регіонах] - Режим доступу до ресурсу:https://briarproject.org/how-it-works/ (Рік звернення 2023)

23. GitLab, briar/briar GitLab [Електронний регіонах] - Режим доступу до ресурсу:https://code.briarproject.org/briar/briar (Рік звернення 2023)

24. Google, Briar - Додатки в Google Play [Електронний регіонах] - Режим доступу до ресурсу:https://play.google.com/store/apps/details?id=org.briarproject.briar.android&hl=uk&gl=US (Рік звернення 2023)

25. Briar, Secure messaging, anywhere - Briar [Електронний регіонах] - Режим доступу до ресурсу:https://briarproject.org/ (Рік звернення 2023)

26. Ryan Robinett, Tiago Royer, Encoding SPAN Evolution Using Frequent-Collision Blockchains // University of Chicago 2019

27. IBM, Blockchain for digital identity and credentials [Електронний ресурс] - Режим доступу до ресурсу:https://www.ibm.com/blockchain-identity (Рік звернення 2023)

28. Gupta, Manav, Blockchain For Dummies®, 2nd IBM Limited Edition // John Wiley & Sons, Inc, 2018

29. Amazon AWS, What is Blockchain Technology? - Blockchain Technology Explained - AWS [Електронний ресурс] - Режим доступу до ресурсу:https://aws.amazon.com/what-is/blockchain/ (Рік звернення 2023)

30. Forbes, Five Non-Financial Blockchain Use Cases Marketers Need To Understand [Електронний ресурс] - Режим доступу до ресурсу:https://www.forbes.com/sites/forbescommunicationscouncil/2018/03/27/five-non-financial-blockchain-use-cases-marketers-need-to-understand (Рік звернення 2023)

31. GAO, Blockchain: Financial and Non-Financial Uses and Challenges [Електронний ресурс] - Режим доступу до ресурсу:https://www.gao.gov/blog/blockchain-financial-and-non-financial-uses-and-challenges (Рік звернення 2023)

32. Medium, Exploring non-financial use cases of blockchain [Електронний ресурс] - Режим доступу до ресурсу:https://medium.com/district3/exploring-non-financial-use-cases-of-blockchain-2839bacd50a4 (Рік звернення 2023)

33. Openledger, 9 Real-life Blockchain and IoT Use Cases [Електронний ресурс] - Режим доступу до ресурсу:https://openledger.info/insights/blockchain-iot-use-cases/ (Рік звернення 2023)

34. IBM, How does IoT work with blockchain?[Електронний ресурс] - Режим доступу до ресурсу:https://www.ibm.com/topics/blockchain-iot (Рік звернення 2023)

35. Harner, Isabel, 3 Ways IoT Could Save the Environment, 2018 [Електронний ресурс] - Режим доступу до ресурсу:https://www.iotforall.com/iot-environment-conservation-poaching/ (Рік звернення 2023)

36. Pauw, Chrisjan, How Significant Is Blockchain in Internet of Things? // Cointelegraph, 2018 [Електронний ресурс] - Режим доступу до ресурсу:https://cointelegraph.com/news/how-significant-is-blockchain-in-internet-of-things (Рік звернення 2023)

37. IOTA, What is IOTA | IOTA [Електронний ресурс] - Режим доступу до ресурсу:https://www.iota.org/get-started/what-is-iota (Рік звернення 2023)

38. Deloitte, Can blockchain accelerate Internet of Things (IoT) adoption | Deloitte Switzerland [Електронний ресурс] - Режим доступу до ресурсу:https://www2.deloitte.com/ch/en/pages/innovation/articles/blockchain-accelerate-iot-adoption.html (Рік звернення 2023)

39. Daley, Sam, 10 Key Blockchain IoT Companies You Should Know | Built In [Електронний ресурс] - Режим доступу до ресурсу:https://builtin.com/blockchain/blockchain-iot-examples (Рік звернення 2023)

40. Chain of Things, Chain of Things [Електронний ресурс] - Режим доступу до ресурсу:https://www.chainofthings.com/ (Рік звернення 2023)

41. Moeco, Data Generation for Enterprise [Електронний ресурс] - Режим доступу до ресурсу:https://moeco.io (Рік звернення 2023)

42. Buck, Jon, New Partnership For Global IoT Connectivity, 2017 [Електронний ресурс] - Режим доступу до ресурсу:https://cointelegraph.com/news/new-partnership-for-global-iot-connectivity (Рік звернення 2023)

43. FOAM, FOAM [Електронний ресурс] - Режим доступу до ресурсу:https://foam.space (Рік звернення 2023)

44. Powerledger, Powerledger Energy Projects [Електронний ресурс] - Режим доступу до ресурсу:https://www.powerledger.io/ (Рік звернення 2023)

45. Berman, Ana, Australian Real Estate Major to Trial Blockchain-Powered Solar Energy Management Solution // Cointelegraph, 2018 [Електронний ресурс] - Режим доступу до ресурсу:https://cointelegraph.com/news/australian-real-estate-major-to-trial-blockchain-powered-solar-energy-management-solution (Рік звернення 2023)

46. MetaMask, MetaMask - Офіційна документація [Електронний ресурс] - Режим доступу до ресурсу:https://docs.metamask.io/ (Рік звернення 2023)

47. WalletConnect, WalletConnect - Офіційна документація [Електронний ресурс] - Режим доступу до ресурсу:https://docs.walletconnect.com/2.0/ (Рік звернення 2023)

48. Moralis, Moralis Auth API - Офіційна документація [Електронний ресурс] - Режим доступу до ресурсу:https://docs.moralis.io/authentication-api (Рік звернення 2023)

49. EIP, ERC-4361: Sign-In with Ethereum - Опис стандарту [Електронний ресурс] - Режим доступу до ресурсу:https://eips.ethereum.org/EIPS/eip-4361 (Рік звернення 2023)

50. GitHub, Репозиторій проекту в роботі [Електронний ресурс] - Режим доступу до ресурсу:https://github.com/Lionel-Logue/SPAN-Blockchain-Auth-App (Рік звернення 2023)

51. Богдан Карлаш, Богдан Овсак, Юрій Бойко, Features of using IoT in higher education institution // The 18th INTERNATIONAL CONFERENCE ON ELECTRONICS AND APPLIED PHYSICS, Київ, 18-22 жовтня, 2022 р.

52. Богдан Карлаш, Леонід Чепель, Юрій Бойко, Comparative Analysis of Smartphone Ad-Hoc Network Based Software // XXIII International Young Scientists Conference on Applied Physics, Київ, 16-20 травня, 2023 р.

ДОДАТКИ

Додаток А Код кнопки Sign In

```
public void signIn(View view){
    EditText profileId_tv = findViewById(R.id.profileID); //user's profile id
from respective field
    EditText address_tv = findViewById(R.id.address); //user's address from
respective field
    String apiKey =
"z3xTSDevCaLveVMLQkDuOWvPvLiOHGtva08EWEzhzBuaU2xHMeZetamqVZel1PKe"; //Moralis
api key, required to use the Api

    OkHttpClient client = new OkHttpClient(); //a tool that allows to perform
http requests for verification of profile id,
    //it requires implementation 'com.squareup.okhttp3:okhttp:4.10.0' in app
gradle file and need imports

    //the following request calls https://authapi.moralis.io/ and asks for
information on the given profile id using given api key,
    //if the user exists, it returns the user's address
    Request request = new Request.Builder()
            .url("https://authapi.moralis.io/profile/" + profileId_tv.getText()
+ "/addresses")
            .addHeader("accept", "application/json")
            .addHeader("X-API-Key", apiKey)
            .build();
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onFailure(Call call, IOException e) {
            Toast.makeText(getApplicationContext(), "Error: "+e.toString(),
Toast.LENGTH_SHORT).show(); //throws exception on error
        }

        @Override
        public void onResponse(Call call, Response response) throws IOException
{
            // Handle response
            final String responseBody = response.body().string(); //the response
given by https://authapi.moralis.io/

            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    //looks if the given response contains provided address, if
there is no address corresponding to the given one, throws an error,
                    //if it exists, the user is logged in
                    if(responseBody.contains(address_tv.getText())){
                        Toast.makeText(getApplicationContext(), "Successful!",
Toast.LENGTH_SHORT).show();
                        setContentView(R.layout.logged_in);
                    }
                    else{
                        Toast.makeText(getApplicationContext(), "Error: No such
Address found!", Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
    });
}
```

Додаток Б Файл .env.local

```
MORALIS_API_KEY= 'z3xTSDevCaLveVMLQkDuOWvPvLiOHGtva08EWEzhzBuaU2xHMeZetamqVZel1PKe'
APP_DOMAIN=10ca-213-111-70-241.ngrok-free.app
NEXTAUTH_URL=https://10ca-213-111-70-241.ngrok-free.app
NEXTAUTH_SECRET=0e181b7caba005381e8297c8ed9943a0eb493e111f1486f1c7ef920f29afb50d
```

Додаток В Файл index.tsx

```
import { MetaMaskConnector } from 'wagmi/connectors/metaMask';//nedded to connect
to MetaMask
import { signIn } from 'next-auth/react';//used to allow sign in flow by
redirecting upon seccessful sign in
import { useAccount, useConnect, useSignMessage, useDisconnect } from
'wagmi';//needed to interact with MetaMask wallet
import { useRouter } from 'next/router';//allows navigation between pages
import { useAuthRequestChallengeEvm } from '@moralisweb3/next';//the Moralus hook
for requesting a challenge message for authentication

//This defines a component that renders a button with an event listener that calls
the function handleAuth() when clicked.
function SignIn() {
    const { connectAsync } = useConnect();
    const { disconnectAsync } = useDisconnect();
    const { isConnected } = useAccount();
    const { signMessageAsync } = useSignMessage();
    const { requestChallengeAsync } = useAuthRequestChallengeEvm();
    const { push } = useRouter();

    //This function first checks if there is an active connection and disconnects
if there is one.
    //Then it connects to MetaMask using the imported connector and requests a
challenge message for authentication.
    //After signing this message, it calls the imported signIn function and
redirects to ‘/user’ page.
    const handleAuth = async () => {
        if (isConnected) {
            await disconnectAsync();
        }

        const { account, chain } = await connectAsync({ connector: new
MetaMaskConnector() });

        const { message } = await requestChallengeAsync({ address: account,
chainId: chain.id });

        const signature = await signMessageAsync({ message });

        // redirect user after successful authentication to '/user' page
        const { url } = await signIn('moralis-auth', { message, signature,
redirect: false, callbackUrl: '/user' });
```

```
        push(url);
    };

    return (
        <div>
            <h3>Web3 Authentication</h3>
            <button onClick={handleAuth}>Authenticate via Metamask</button>
        </div>
    );
}

export default SignIn;
```

Додаток Г Файл user.jsx

```
import { getSession, signOut } from 'next-auth/react';
//getSession function retrieves the session object for the current user.
//The session object contains information about the user's authentication status
and any additional data that was returned by the authentication provider.
//The signOut redirects the user back to main page
//JSON returns the information about the user session including profileId, address
and other parameters
function User({ user }) {
    return (
        <div>
            <h4>User session:</h4>
            <pre>{JSON.stringify(user, null, 2)}</pre>
            <button onClick={() => signOut({ redirect: '/' })}>Sign out</button>
        </div>
    );
}

export async function getServerSideProps(context) {
    const session = await getSession(context);

    if (!session) {
        return {
            redirect: {
                destination: '/',
                permanent: false,
            },
        };
    }

    return {
        props: { user: session.user },
    };
}

export default User;
```

Додаток Ґ Файл [...nextauth].ts

```
import NextAuth from 'next-auth';
import { MoralisNextAuthProvider } from '@moralisweb3/next';

export default NextAuth({
    providers: [MoralisNextAuthProvider()],
    // adding user info to the user session object
    callbacks: {
        async jwt({ token, user }) {
            if (user) {
                token.user = user;
            }
            return token;
        },
        async session({ session, token }) {
            (session as { user: unknown }).user = token.user;
            return session;
        },
    },
});
```

Додаток Д Файл [...moralis].ts

```
import { MoralisNextApi } from '@moralisweb3/next';

export default MoralisNextApi({
    apiKey: process.env.MORALIS_API_KEY,
    authentication: {
        domain: process.env.APP_DOMAIN,
        uri: process.env.NEXTAUTH_URL,
        timeout: 120,
    },
});
```