Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Комплексна бакалаврська дипломна робота на тему:

«Засіб захищеної інтернет-телефонії. Частина 1. Модуль автентифікації сторін»
08-20.БДР.014.00.000

Виконав: студент 4 курсу групи 1БС-19б
спеціальності 125 Кібербезпека
_____ Дмитро СОКОЛ

Керівник: к. т. н., доцент, доцент каф. ЗІ
_____ Юрій БАРИШЕВ
«16» червня 2023 р.

Рецензент: к. т. н., доцент, доцент каф. ПЗ
_____ Наталя БАБЮК
«16» червня 2023 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
«16» червня 2023 р.

Вінниця ВНТУ – 2023 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д. т. н., професор
_____ Володимир ЛУЖЕЦЬКИЙ
«20» березня 2023 року

З А В Д А Н Н Я
НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Соколу Дмитру Анатолійовичу

1. Тема роботи: «Засіб захищеної інтернет-телефонії. Частина 1. Модуль автентифікації сторін»
керівник роботи: Баришев Юрій Володимирович, к. т. н., доцент, доцент кафедри ЗІ,
затверджено наказом ректора ВНТУ від 20 березня 2023 року №67.
2. Строк подання студентом роботи 17 червня 2023 р.
3. Вихідні дані до роботи:
   – встановлення захищеного зв'язку;
   – попередній аналіз методі передачі мультимедіа даних;
   – програмна реалізація методів;
4. Зміст текстової частини: Вступ. 1. Аналіз методів захисту зв'язку в інтернет-телефонії. 2. Архітектура модуля захищеного зв'язку. 3. Алгоритми роботи модуля захищеного зв'язку. 4. Тестування засобу захищеної інтернет-телефонії. Висновки. Список використаних джерел. Додатки.

5. Перелік ілюстративного матеріалу: порівняльний аналіз засобів інтернет-телефонії (плакат А4), аналіз криптографічних протоколів автентифікації (плакат А4), архітектура засобу захищеної інтернет-телефонії (плакат А4), структура модулю автентифікації сторін (плакат А4), узагальнений алгоритм роботи засобу (плакат А4), алгоритм роботи блоку автентифікації сторін (плакат А4), алгоритм роботи блоку автентифікації користувачів (плакат А4), результати блокового тестування (плакат А4), результати інтеграційного тестування (плакат А4)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийня
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	10.04.23
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.03.23	09.05.23
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	21.05.23
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ	20.05.23	10.06.23

7. Дата видачі завдання 20 березня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

ч	Назва етапів комплексної бакалаврської дипломної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	20.03.23 – 26.03.23	
2	Аналіз джерел за напрямком комплексної бакалаврської дипломної роботи	27.03.23 – 09.04.23	
3	Розробка архітектури та алгоритму засобу	10.04.23 – 23.04.23	
4	Програмна реалізація та тестування	24.04.23 – 21.05.23	
5	Аналіз результатів тестування, висновки	22.05.23 – 24.05.23	
6	Оформлення пояснювальної записки	25.05.23 – 31.05.23	
7	Попередній захист та доопрацювання БДР	01.06.23 – 15.06.23	
8	Представлення БДР на захист	16.06.23 – 19.06.23	
9	Захист БДР	20.06.23 – 23.06.23	

Студент _____ Дмитро СОК

Керівник роботи _____ Юрій БАРИШ

АНОТАЦІЯ

Комплексна бакалаврська дипломна робота складається з 55 сторінок формату А4, на яких є 14 рисунків, 5 таблиць, список використаних джерел містить 22 найменувань.

Комплексна бакалаврська роботи присвячена розробці програмного засобу для реалізації захищеної інтернет-телефонії. В результаті аналізу аналогічних засобів, методів та криптографічних протоколів автентифікації було розроблено власний криптографічний протокол автентифікації на основі односпрямованої геш-функції SHA-3-256 та псевдовипадкових чисел. Даний протокол дозволяє реалізувати двосторонню автентифікації віддалених користувачів. Обраний криптографічний алгоритм забезпечує стійкий захист даних користувачів від несанкціонованої зміни чи порушення конфіденційності. Розроблено архітектуру засобу та алгоритми роботи засобу, розроблено програмну реалізацію модуля та виконано його тестування.

Ключові слова: захищена інтернет-телефонія, автентифікація, методи автентифікації, гешування, SHA-3-256.

ABSTRACT

The comprehensive bachelor thesis consists of 55 pages of A4 format, on which there are 14 figures, 5 tables, the list of used sources contains 22 titles.

The complex bachelor's thesis is devoted to the development of a software tool for the implementation of secure Internet telephony. As a result of the analysis of similar means, methods and cryptographic authentication protocols, an own cryptographic authentication protocol was developed based on the one-way hash function SHA-3-256 and pseudo-random numbers. This protocol allows you to implement two-way authentication of remote users. The selected cryptographic algorithm ensures stable protection of user data against unauthorized changes or privacy violations. The tool architecture and tool operation algorithms were developed, the software implementation of the module was developed and its testing was performed.

Keywords: secure Internet telephony, authentication, authentication methods, hashing, SHA-3-256.

ЗМІСТ

ВСТУП

В двадцять першому столітті, через різноманітні перепони та загрози більшість людей змінила звичний формат живого спілкування на використання засобів інтернет-телефонії, ці засоби щоденно використовуються великою кількістю людей по всьому світу, оскільки вони мають набагато ширший ряд можливостей, в порівнянні зі звичайним зв'язком шляхом телефонії, тим більше, з форматом живого спілкування, якому перешкоджають певні сучасні проблеми. За значним поширенням та розвитком, даних засобів, з’явились і проблеми забезпечення безпеки конфіденційних даних, що циркулюють в таких засобах.

Популярні, засоби інтернет-телефонії належним чином не достатньо піклуються про забезпечення конфіденційності даних, як на етапах автентифікації, так і при наступному їх зберіганні та передаванні. Це можна пояснити нагальною потребою в таких засобах і поспіхом розробників вийти на ринок з продуктом якомога раніше. Відповідно питання безпеки не було в їх фокусі в гонитві за першістю в заповненні ринку, який неочікувано стрімко виріс через пандемію, тому питання кібербезпеки розглядались як другорядні [1]. Також варто відзначити, неодноразові докази того, що компанії збирають через такі засоби велику кількість конфіденційних даних, від номерів телефонів до біометричних особливостей користувача, таких як відбитки пальців, і всі ці дані збираються через використання сумнівних методів автентифікації. Саме тому актуально розробити засіб, який би не використовував конфіденційні дані користувачів такі як біометрію та номери телефону під час автентифікації користувачів.

Об’єктом даного дослідження є процес обміну аудіо та відео даними.

Предметом - методи та криптографічні протоколи автентифікації.

Метою даної комплексної бакалаврської дипломної роботи є покращення захисту медіа даних користувачів під час користування засобами інтернет-телефонії.

Для досягнення мети потрібно розв’язати такі задачі:

– проаналізувати засоби інтернет-телефонії

- проаналізувати відомі методи автентифікації
- проаналізувати відомі криптографічні протоколи автентифікації
- розробити архітектуру модуля автентифікації засобу захищеної інтернет-телефонії
- розробити алгоритми роботи модуля автентифікації;
- реалізувати алгоритми та виконати тестування розробленого засобу.

Результати, отримані під час бакалаврського кмплексного дипломного проектування доповідались на - LII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2023) та були опубліковані - Баришев Ю.В., Сокол Д.А. ЗАСІБ ЗАХИЩЕНОГО АУДІО ТА ВІДЕО ЗВ’ЯЗКУ // Матеріали тезів LII наук.-техн. конф. ФІТКІ, ВНТУ. Вінниця. 2023 – 2 С. (на програмного комітету)

1 АНАЛІЗ МЕТОДІВ АВТЕНТИФІКАЦІЇ СТОРІН В ІНТЕРНЕТ-ТЕЛЕФОНІЇ

1.1 Аналіз засобів інтернет-телефонії

Наразі розроблено чимало програмного забезпечення для встановлення аудіо- та відео-зв’язку, які пропонують різні методики захисту даних. Загалом слід розглядати дані застосунки під призмою їх функціоналу, наприклад встановлення аудіо- та відео-зв’язку, методів захисту даних, включаючи методи автентифікації сторін, криптографічних протоколів автентифікації та шифрування.

Для початку варто розглянути найрозповсюдженіший програмний засіб захищеного аудіо- та відео-зв’язку під назвою WhatsApp від компанії Meta [2]. Даний програмний засіб є найстаршим серед приведених нижче аналогів, що обумовило його високу популярність та попри те, що він має доволі стандартний функціонал, через свій вік WhatsApp пройшов через велику кількість проблем з безпекою та захистом. Серед його функціоналу є такі функції, як: надсилання текстових, відео, аудіо повідомлень, встановлення аудіо та відео зв’язку, створення групових чатів та каналів. WhatsApp використовує наскрізне шифрування для всіх розмов, що захищає повідомлення від перехоплення третіми сторонами під час передачі. WhatsApp забезпечує двоетапну перевірку для додаткової безпеки. З квітня 2016 з виходом оновлення версії 2.16.12 WhatsApp включив наскрізне шифрування (end-to-end) для всіх користувачів на базі розробок протоколу Signal. Цей протокол використовує комбінацію протоколу узгодження ключів X3DH[3], алгоритму Double Ratchet для безпечного керування ключами та 256-бітного симетричного шифрування AES разом із обміном ключами за протоколом Діффі-Геллмана на основі еліптичних кривих (ECDH) для шифрування повідомлень. Шифрування поширюється на всі типи повідомлень: текст, фото, відео та голосові повідомлення. Шифрування також доступне у групових чатах. WhatsApp використовує різні методи автентифікації для перевірки та підтвердження ідентичності користувачів. Основні види автентифікації, які використовуються в WhatsApp, включають:

- геш-код - після реєстрації WhatsApp генерує геш-код на основі номера телефону користувача та інших даних. Цей геш-код використовується для автентифікації користувача при кожному вході в застосунок. Геш-код зберігається на пристрої користувача та використовується для перевірки його ідентичності.
- номер телефону - WhatsApp прив'язаний до номера мобільного телефону користувача. При реєстрації в WhatsApp користувач повинен підтвердити свій номер телефону шляхом отримання SMS-повідомлення або використання автоматичного дзвінка з унікальним кодом підтвердження. Це дозволяє підтвердити, що користувач має доступ до вказаного номера телефону.
- QR-код - WhatsApp також підтримує автентифікацію за допомогою QR-кодів. Користувач може сканувати QR-код на пристрої, що вже автентифікований в WhatsApp, для з'єднання свого облікового запису з цим пристроєм. Це забезпечує зручність та швидкість процесу автентифікації.
- біометричні дані - WhatsApp підтримує використання біометричних даних, таких як сканування відбитка пальця або розпізнавання обличчя, для автентифікації та розблокування доступу до програми. Це додатковий рівень безпеки, який забезпечує, що лише власник пристрою має доступ до облікового запису WhatsApp.
- використання декількох факторів автентифікації - WhatsApp підтримує можливість використання двофакторної автентифікації (2FA). Користувачі можуть налаштувати додатковий пароль або PIN-код, який потрібно буде ввести під час першого входу на новий пристрій або після тривалої неактивності. Це додає додатковий шар захисту для облікового запису користувача. До недоліків можна віднести те, що WhatsApp - підтримує політику обміну даними з материнською компанією Facebook, що може призвести до використання даних користувачів для цільової реклами та інших цілей. Також WhatsApp не має традиційної функції «вийти», що означає, що користувачі завжди перебувають в обліковому записі

застосунку, що ускладнює певним чином користування.

Наступним варіантом аналогу для аналізу можна розглянути програмний засіб Skype від компанії розробника Microsoft [4]. Skype використовує протокол, що передбачає шифрування трафіку, TLS (Transport Layer Security)[5] для захисту приватності та конфіденційності даних, переданих під час розмови. TLS забезпечує криптографічний захист шляхом шифрування даних, що передаються між клієнтами Skype і серверами компанії Microsoft. Це допомагає унеможливити прослуховування та підслуховування комунікації третіми особами, також використовується 256-розрядний AES над 128-розрядними блоками, який. Skype використовує алгоритми гешування, наприклад, SHA-256 (Secure Hash Algorithm 256-bit), для створення гешів інформації, таких як паролі або контрольні суми. Гешування допомагає забезпечити цілісність та перевірку даних під час передачі.

Щодо автентифікації, Skype використовує різні методи автентифікації для перевірки та підтвердження ідентичності користувачів. Основними видами автентифікації, що використовуються в Skype, є такі [4]:

- парольна - користувачі мають створювати облікові записи з унікальними ідентифікаторами (ім'я користувача або електронна пошта) та паролями. Цей метод є найпоширенішим і забезпечує перевірку ідентичності на основі збігу введених та еталонних логіну та пароля.
- багатофакторна автентифікація - Skype підтримує функцію двоетапної перевірки (2FA). Це дозволяє користувачам налаштувати додатковий шар безпеки, запитуючи додатковий код підтвердження після введення основного пароля. Код може бути надісланий через SMS або генеруватися додатком автентифікації.
- прив’язка до облікового запису Microsoft - Skype пов'язаний з обліковим записом Microsoft, що дозволяє використовувати його як фактор автентифікації. Це може включати використання Windows Hello або інших біометричних методів, таких як відбиток пальця або розпізнавання обличчя.
- біометрична - також підтримується використання відбитка пальця або

розпізнавання обличчя для автентифікації користувача.

Останнім програмним засобом, який буде розглянуто, буде Signal від розробника Signal Foundation [6], даний аналог має подібний функціонал, а саме захищені текстові, відео- та аудіоповідомлення, можливість встановлення аудіо- та відеозв'язку, створення групового чату та встановлення режиму зникнення повідомлень через обраний період часу. Signal використовує протокол Signal Protocol, який використовує наскрізне шифрування і був розроблений спеціально для безпечного обміну миттєвими повідомленнями. Протокол сигналу використовує комбінацію протоколу узгодження ключів X3DH і алгоритму Double Ratchet[7] для безпечного керування ключами, а також 256-бітове симетричне шифрування AES [8] і обмін ключами за проптоколом Діффі-Геллмана на основі еліптичних кривих (ECDH) для шифрування повідомлень. Методи автентифікації, що використовуються в Signal:

- номер телефону - Signal використовує номер телефону користувача як основний ідентифікатор. При реєстрації в Signal, користувач повинен підтвердити свій номер телефону, а також створити і підтвердити код реєстрації, що надсилається через SMS.
- багатофакторна автентифікація - Signal підтримує можливість включення двофакторної автентифікації для додаткового рівня безпеки. При цьому користувач повинен налаштувати додатковий пароль, який буде використовуватися під час входу в додаток або відновлення облікового запису.
- сертифікація ключів - Signal дозволяє користувачам перевіряти "відбиток" ключа для підтвердження ідентичності особи під час обміну повідомленнями.

Signal не має стільки функцій налаштувань, як інші застосунки, такі як WhatsApp, також він не має функції групового дзвінка, хоч Signal і більше зосереджений на конфіденційності, ніж інші застосунки захищеного аудіо та відео зв'язку, він покладається на централізовані сервери, які потенційно можуть бути зламані, скомпрометовані.

В результаті проведення попереднього аналізу засобів інтернет-телефонії, далі було порівняно та проаналізовано вищезазначені засоби інтернет-телефонії, в таблиці 1.1 за такими характеристикам: підтримка апаратних засобів, мультиплатформеність (Операційні Системи), шифрування, анонімність, збереження даних, інтеграція з іншими сервісами. Окремо було наведено порівняння використовуваних методів автентифікації в таблиці 1.2.

Таблиця 1.1 – Порівняльний аналіз проаналізованих засобів

Характеристика	Signal	WhatsApp	Skype
Підтримка апаратних засобів	Немає	Немає	Фізичний ключ YubiKey
Мультиплатформеність(Операційні Системи)	iOS, Android, Microsoft Windows, Linux, macOS	Android, iOS, Microsoft Windows macOS, S40, Tizen, KaiOS	Microsoft Windows, macOS, Android, iOS, Symbian OS, Windows Phone, Linux, BlackBerry OS, webOS
Шифрування	Шифрування на стороні користувача, застосовується E2E шифрування	Шифрування на стороні користувача, застосовується E2E шифрування	Шифрування на стороні користувача але E2E шифрування не застосовується для всіх типів повідомлень
Анонімність	Анонімний	Не анонімний, потрібен номер телефону	Не анонімний, потрібен, обліковий запис Microsoft
Збереження даних	Зберігаються на серверах, після доставки повідомлення вони видаляються з сервера, не зберігає метадані	Зберігаються на серверах, можливе видалення повідомлень з обох сторін	Зберігаються на серверах, можливе видалення повідомлень з обох сторін
Інтеграція з іншими сервісами	Обмін файлами, можливість інтеграції з деякими сервісами	Обмін файлами інтеграція з різними сервісами, такими як iCloud	Обмін файлами, інтеграція з різними сервісами, такими як OneDrive

Таблиця 1.2 – Порівняльний аналіз використання методів автентифікації в

засобах

Методи автентифікації	Signal	WhatsApp	Skype
Парольна	+	+	+
2FA	+	+	+
Біометрична	-	+	+
Одноразові паролі(OTP)	+	+	+
OpenID Connect	-	-	+

Додатково проаналізувавши вищезазначені програмні засоби було визначено, що загальною проблемою застосунків є залежність від номера телефону, WhatsApp та Signal дуже сильно покладаються на реєстрацію за номером телефону, проблема заключається в тому, що сам по собі індентифікатор в вигляді номеру телефону є не дуже анонімним, присутня можливість заволодіння номером небажаними користувача, що може привести до цілком незручних ситуацій в житті, також при втраті телефону користувач втрачає абсолютно всі дані та аккаунт як такий(за умови, що аккаунт не поширено на інші пристрої).

1.2 Аналіз методів автентифікації сторін

Методи автентифікації сторін були розроблені для забезпечення безпеки та ідентифікації користувачів у різних системах. Головною метою автентифікації є переконання в тому, що особа або сутність, яка намагається отримати доступ до системи, дійсно є тим, за кого вона себе видає.

Загально можна класифікувати методи автентифікації сторін наступним чином:

– засновані на знаннях;

– засновані на володінні;

– засновані на властивостях.

Методи, засновані на знаннях поділяються на такі:

– пароль - вимагається введення секретного пароля, який повинен бути

відомий тільки автентифікованій стороні;

– питання для відновлення - вимагається відповідь на питання, яке користувач встановив раніше, щоб підтвердити свою ідентичність;
– одноразовий пароль - користувач отримує одноразовий код або пароль, який використовується лише один раз для автентифікації.

Методи, засновані на володінні поділяються на такі:

– фізичний токен - фізичний пристрій, такий як смарт-карта або USB-ключ, який містить інформацію, необхідну для автентифікації;
– RFID-картки – широко використовувані безконтактні пристрої побудовані на інтегральних схемах, в вигляді карти, що зберігає унікальний ідентифікатор;
– імплантований пристрій – зазвичай мікрочіп, що використовується для автентифікації особи шляхом вбудовування чіпу в тіло користувача. Цей пристрій може містити ідентифікатори або ключі, які використовуються для підтвердження особи при доступі до систем або послуг.

Методи, засновані на властивостях поділяються на такі:

– геолокаційна автентифікація - використання географічних даних або місцезнаходження для підтвердження особи;
– біометричні дані - використання унікальних фізичних характеристик, таких як відбиток пальця, сканер обличчя або розпізнавання голосу, для підтвердження ідентичності;

Також важливим фактором в контексті аналізу методів автентифікації класифікація методів автентифікації за сторонами, вона включає наступні категорії:

– одностороння автентифікація (One-Way Authentication) - у цьому методі тільки одна сторона, зазвичай клієнт або користувач, підтверджує свою ідентичність перед іншою стороною, як правило, сервером. У цьому випадку тільки одна сторона презентує автентифікаційні дані, такі як пароль або сертифікат, для підтвердження своєї ідентичності. Прикладом є традиційний вхід до аккаунту з використанням пароля.

– двостороння автентифікація (Two-Way Authentication) - у цьому методі обидві сторони, як клієнт, так і сервер, підтверджують свою ідентичність одна перед одною. Це забезпечує взаємну перевірку ідентичності між сторонами. Кожна сторона представляє свої автентифікаційні дані, такі як сертифікати або цифрові підписи, для перевірки своєї ідентичності. Прикладом є взаємна автентифікація, яка відбувається під час з'єднання між клієнтом і сервером у протоколі TLS/SSL.
– багатостороння автентифікація (Multi-Factor Authentication) - у цьому методі залучаються багато сторін для підтвердження ідентичності користувача. Наприклад третя сторона, така як зовнішній постачальник ідентичності або сервіс автентифікації, забезпечує процес перевірки і підтвердження ідентичності користувача. Користувач надає свої облікові дані третій стороні, яка перевіряє їх відповідність ідентичності та видає токен або підтвердження автентифікації, яке може бути використане для доступу до ресурсів або послуг. Протоколи, які використовують автентифікацію за довіреною третьою стороною, включають OAuth, OpenID[9] .

Проаналізувавши класифікацію методів автентифікації за кількістю сторін, можна дійти висновку, що найліпшим варіантом є саме двостороння автентифікація, оскільки у двосторонньому методі обидві сторони, як до прикладу Аліса, так і Боб, перевіряють автентичність один одного. Це дозволяє уникнути атак, які спрямовані на підробку ідентичності однієї зі сторін та забезпечує довіру між ними, ні одностороння ні багатостороння такої довіри забезпечити не може, оскільки багатостороння як мінімум спирається на безперечну довіру наприклад третій стороні, що може бути скомпрометована, а тому не може бути використана в межах цієї комплексної бакалаврської дипломної роботи.

Аналіз методів автентифікації сторін включає огляд різних способів і засобів, які використовуються для перевірки та підтвердження автентичності користувачів під час взаємодії з веб-сторінками. Основна мета автентифікації сторін полягає у переконанні однієї сторони в ідентичності іншої, тобто переконанні, в тому, що

сторона дійсно являється тою, за яку вона себе видає. В контексті аналізу методів автентифікації сторін варто також класифікувати протоколи автентифікації, тому нижче наведено класифікацію протоколів автентифікації сторін за рівнем безпеки:

- проста автентифікація;
- сувора автентифікація;
- протоколи доведення з нульовим розголошенням знань.

Протоколи, засновані на простій автентифікації поділяються на такі:

- парольна - користувачі мають створювати облікові записи з унікальними ідентифікаторами (ім'я користувача або електронна пошта) та паролями. Цей метод є найпоширенішим і забезпечує перевірку ідентичності на основі збігу введених та еталонних логіну та пароля.
- одноразові паролі (OTP) - при вході на новий пристрій або в разі втрати доступу до облікового запису Telegram може надіслати користувачу одноразовий пароль (OTP) через SMS або пошту для відновлення доступу.
- JSON Web Token (JWT) - це відкритий стандарт (RFC 7519) для створення токенів доступу, що базується на форматі JSON. Як правило, використовується для передачі даних для автентифікації в клієнт-серверних програмах. Токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує цей токен для підтвердження справжності облікового запису [9].
- геш-код - під час реєстрації, пароль користувача проходить через геш-функцію, і результат (хеш-код) зберігається в базі даних або системі авторизації. При спробі входу, користувач надає свій ідентифікатор та пароль. Система обчислює геш-код введеного паролю і порівнює його зі збереженим геш-кодом. Якщо значення збігаються, користувача вважають автентифікованим.
- біометрична автентифікація - цей метод використовує фізичні характеристики користувача, такі як відбитки пальців, розпізнавання обличчя, сканування сітківки ока, для підтвердження автентичності. Біометрична автентифікація може забезпечити високий рівень безпеки,

оскільки фізичні характеристики важко підробити або скопіювати.

- використання токенів доступу - цей метод використовується для автентифікації веб-сторінок, які пропонують API або доступ до зовнішніх ресурсів. Користувач отримує токен доступу після успішної автентифікації, який використовується для подальшого доступу до ресурсів. Токени можуть бути обмежені за часом дії та обсягом дозволів, що робить можливою більшу гнучкість та безпеку управління доступом.
- багатофакторна автентифікація (MFA) - цей метод використовує комбінацію двох або більше факторів для підтвердження ідентичності користувача. Цей може включати будь-які вищевказані методи. MFA забезпечує більш високий рівень безпеки порівняно зі звичайним паролем.

Проаналізувавши протоколи засновані на простій автентифікації, можна дійти висновку, що безумовно найкращою в плані безпеки є використання Багатофакторної автентифікації (MFA), оскільки вона може істотно зменшити імовірність викрадення особистих даних в інтернеті, тому, що знання пароля жертви недостатньо для здійснення шахрайства. Наприклад для формування модулю автентифікації клієнт-сервер, можна використати метод геш-функції, оскільки геш-функція є односпрямованою, тобто вирахувати з геш-коду початкове значення паролю, за умови використання криптографічно-стійкої геш-функції, як наприклад SHA-3 [10], математично дуже важко особливо в рамках часу автентифікації, при реєстрації та потім для автентифікації користувача, після цього, використати метод JWT-токену, який буде формуватись після проходження першої лінії оборони в вигляді геш-функції, та до прикладу встановити йому час життя годину для подальшого використання в другому етапі проходження автентифікації на сервері.

Для формування ж модулю автентифікації клієнт-клієнт, потрібно розглядати вже криптографічні протоколи та бажано взаємної автентифікації, оскільки рекомендовано, щоб при спробі наприклад дзвінку, обидва користувача були автентифіковані один перед одним, особливо в умовах постійних спроб зловмисників втрутитись в автентифікацію клієнт-клієнт та заволодіти

конфіденційними даними користувачів, криптографічні протоколи забезпечать і стійкість і взаємну автентифікацію. Тому в рамках наступного підрозділу буде проведено аналіз більш стійких, а саме криптографічних протоколів автентифікації.

1.3 Аналіз криптографічних протоколів автентифікації

Існує чимало криптографічних протоколів автентифікації, кожен з них розроблений для своєї конкретної мети. В умовах аналізу криптографічних протоколів автентифікації, для початку було б доречно привести їх класифікацію.

Протоколи, засновані на суворій автентифікації, тобто побудовані на використанні криптографічних алгоритмів і засобів, поділяються на такі [11]:

- побудовані на симетричних алгоритмах шифрування
- побудовані на асиметричних алгоритмах шифрування
- побудовані на основі алгоритмах електронного цифрового підпису
- побудовані на основі однонаправленої геш-функції

Для початку можна розглянути криптографічний протокол автентифікації, з категорії побудованих на основі одно-направленої геш-функції. Для прикладу буде розглянуто протокол взаємної автентифікації MS-CHAPv2 [12], що використовує DES(алгоритм для симетричного шифрування) та SHA-1 і MD-4[13], для зберігання паролів(алгоритми криптографічного гешування) та механізм «виклик-відповідь». MS-CHAPv2 є протоколом автентифікації, схожим на CHAP, але має важливу відмінність: у CHAP сервер повинен зберігати в оборотно-зашифрованому вигляді пароль(секрет) користувача, який розшифровується при кожній автентифікації клієнта, а в MS-CHAP v2 серверу для цього потрібно тільки геш-значення пароля (секрету). Нижче наведено рисунок, що демонструє алгоритм роботи MS-CHAPv2.

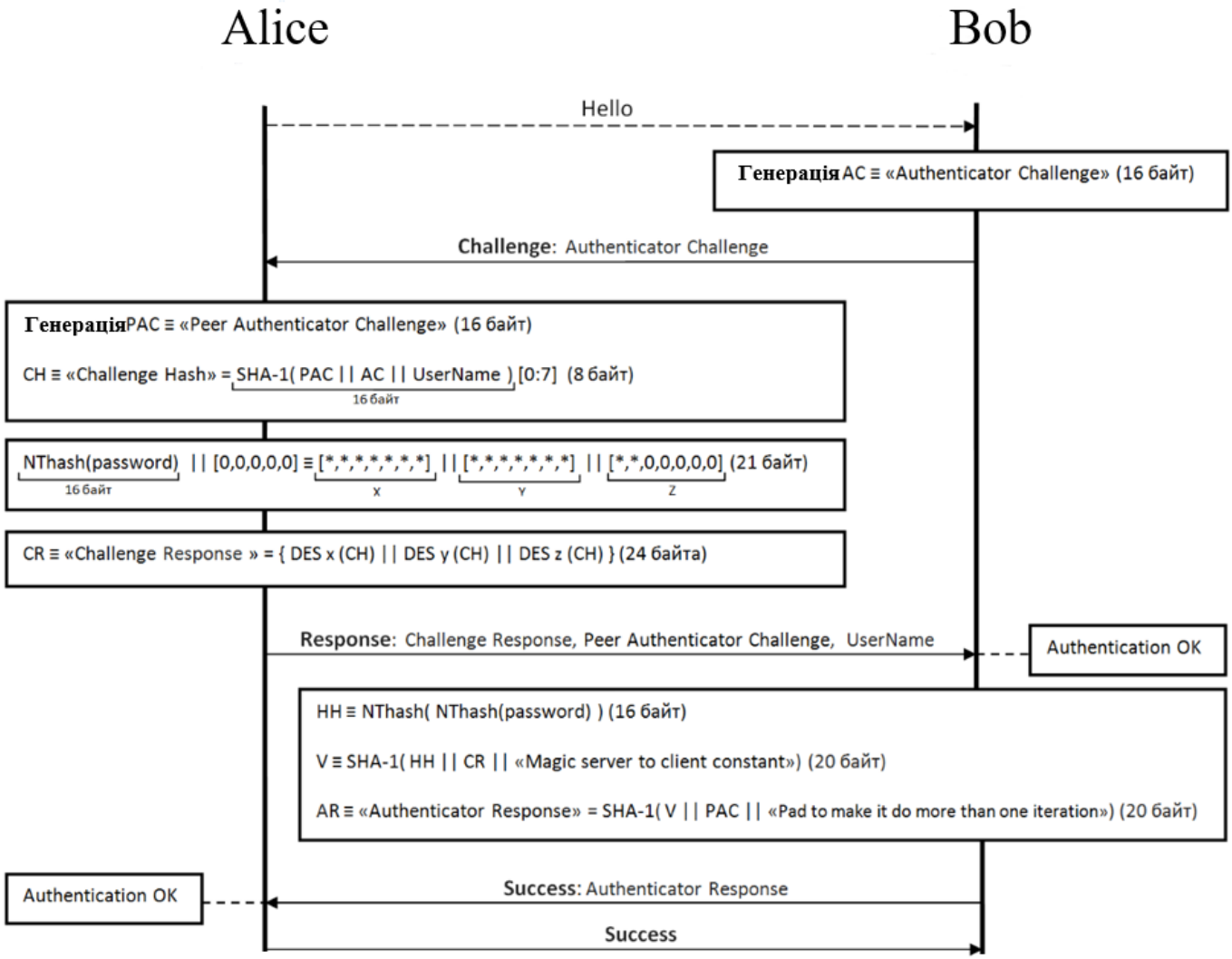


Рисунок 1.1 – Алгоритм роботи протоколу MS-CHAPv2

До загальних недоліків відносяться: MS-CHAPv2 використовує стандартний MD4 алгоритм для гешування паролів. MD4 вважається застарілим і небезпечним з точки зору криптографічних атак, таких як колізійні атаки. Це означає, що теоретично можливо знайти два різні вхідні повідомлення, які генерують однаковий геш-код, цю проблему можна вирішити використовуючи стійкіші геш-функції типу SHA-3 (Secure Hash Algorithm 3). Вразливість до атаки повного перебору DES ключів та перебору словника.

Наступним криптопротоколом можна розглянути, протокол з категорії доказу з нульовим розголошенням знань, наприклад протокол Фейге-Фіата-Шаміра (FFS) [14]. Протокол дозволяє одному учаснику (що доводить А) довести іншому учаснику (що перевіряє В), що він має секретну інформацію, не розкриваючи

жодного біта цієї інформації. А доводить своє знання секрету s стороні В протягом t раундів, не розкриваючи при цьому жодного біта самого секрету. Довірений центр Т публікує велику кількість n=pq, де p і q - прості числа, які тримаються в секреті. Також вибираються цілі числа k і t – параметри безпеки. Кожен учасник обирає k випадкових цілих чисел s1,s2,…sk, 1 ≤ si ≤ n – 1, і k випадкових бітів b1,b2,…bk. Потім обчислює

$$v_i = (-1)^{b_i} * \left(s_i^2\right)^{-1} mod n, 1 \leq i \leq k.$$

Учасник ідентифікує себе оточуючим за допомогою значень (v1,v2,…vk;n) , які виступають як його відкритий ключ, в той час як секретний ключ s = (s1,s2,…sk) відомий лише самому учаснику. Дії протоколу в рамках одного раунду.

1) А вибирає випадкове ціле число r, 1 ≤ r ≤ n-1, обчислє $x = r^2$ mod n, і відправляє x стороні В.
2) В відправляє А випадковий k-бітний вектор (e1,e2,…ek), де ei = 0 або ei = 1.
3) А обчислює та відправляє В: y$r * \prod_{i=1}^{k} {s_i}^{e^i} mod n$.
4) В обчислює $z = y^2 * \prod_{i=1}^{k} {v_i}^{e^i} mod n$ і перевіряє, що z=x mod n $z \neq 0$.

До недоліків даного протоколу можна віднести:

– велике обчислювальне навантаження - протокол FFS вимагає великої кількості обчислень, зокрема, виконання декількох піднесень до степеня та операцій множення. Це може бути проблематичним на мобільних пристроях або при обробці великого обсягу даних.
– потреба у взаємодії з централізованим довіреним сервером - протокол вимагає наявності централізованого довіреного серверу, який відіграє роль посередника між сторонами.

Наступним протоколом, можна розглянути, симетричний протокол Otway-Rees [15]. Протокол Otway-Rees - це симетричний протокол автентифікації, призначений для використання в незахищених мережах (наприклад, Інтернет). Він дозволяє особам, які спілкуються через таку мережу, підтверджувати свою особу

один одному, а також запобігає атакам підслуховування чи повторного відтворення та дозволяє виявити модифікацію. Алгоритм протоколу:

1) Аліса генерує повідомлення з випадкового числа, порядкового номера сесії та ідентифікаторів себе та Боба, після чого шифрує спільним із Трентом ключем. Зашифроване повідомлення разом із порядковим номером та обома ідентифікаторами відправляються Бобу. $Alice\ \ I, A, B, E_A(R_A, I, A, B)Bob.$
2) Боб доповнює це зашифрованими його спільним із Трентом ключем випадковим числом, порядковим номером та ідентифікатором, після чого посилає Тренту — довіреній проміжній стороні. $Bob\ \ I, A, B, E_A(R_A, I, A, B), E_B(R_B, I, A, B)Trent$
3) Трент генерує випадковий сеансовий ключ і надсилає два повідомлення Бобу, разом із номером сесії. Перше зашифровано спільним ключем з Алісою, друге – з Бобом $Trent\ \ I, E_A(R_A, K), E_B(R_B, K)Bob.$
4) Боб отримує ключ і переконується, що випадкове число та порядковий номер сесії не змінилися за час роботи протоколу. $Bob\ \ I, E_A(R_A, K)Alice.$
5) Аліса отримує ключ і переконується, що випадкове число та порядковий номер сесії не змінилися під час роботи протоколу.

До недоліків цього протоколу відноситься, як і в попередніх випадках, використання третьої довіреної сторони також кожен користувач повинен мати свій унікальний ключ для автентифікації. Це може створювати обмеження в масштабованості системи, особливо при великій кількості користувачів.

Всі вищезазначені протоколи, побудовані на різних криптографічних методах, але в контексті аналізу їх можна порівняти за наступними характеристиками: стійкість, швидкодія, ефективність в використанні ресурсів, керування ключами, простота реалізації, підтримка стандартами та масштабованість. Порівняльний аналіз наведено в таблиці 1.3

Таблиця 1.3 – Порівняльний аналіз криптографічних протоколів автентифікації

Характеристика	Протокол Фейге—Фіата—Шаміра	Протокол MS-CHAPv2	Протокол Otway-Rees
Стійкість	Висока	Середня	Середня
Швидкодія	Низька	Помірна	Висока
Ефективність в використанні ресурсів	Середня	Висока	Середня
Простота реалізації	Низька	Середня	Висока
Підтримка стандартами	Ні	Так	Ні
Масштабованість	Середня	Висока	Помірна

Проаналізувавши таблицю, можна зробити наступні висновки, всі представлені протоколи мають свої власні недоліки, протоколи Нідгема-Шредера та Otway-Rees використовують третю довірену сторону, якщо вона тобто сервер припиняє роботу, користувачі не матимуть змоги автентифікуватись, також протокол Нідгема-Шредера без запровадження додаткових заходів, не має захисту від атак повторного використання, протокол Otway-Rees має проблеми з масштабованістю, хоч є найлегшим в простоті реалізації, в рамках невеликої кількості користувачів.

Протокол Фейге-Фіата-Шаміра, має найкращий показник стійкості, але низький показник швидкодії, що в рамках засобу інтернет-телефонії є значним недоліком.

Протокол MS-CHAPv2, має високий показник масштабованості, але програє у всіх інших показниках, порівняно з представленими протоколами, також він потребує додаткових заходів в реалізації оскільки, оригінально налаштований на автентифікацію між сервером та клієнтом, в оригіналі використовує слабкі механізми захисту, такі як – застарілий алгоритм шифрування DES та алгоритми гешування MD-4 [13], SHA-1.

Дані недоліки слід враховувати при обґрунтуванні використаного криптопротоколу для побудови модуля автентифікації користувачів, щоб досягти максимального рівня безпеки, та варто звернути увагу, не тільки, на параметри стійкості, але й на швидкодію та використання ресурсів, також варто розглянути можливість використання власного протоколу, зважаючи, на те , що в рамках аналізу, не було виявлено протоколу, що забезпечив одразу, і стійкість, і швидкодію, для таких умов варто звернути увагу на розробку протоколу, що базується на використанні одно направленої геш-функції.

1.4 Постановка задачі

Базуючись на проаналізованих засобах інтернет-телефонії, методах автентифікації сторін та використаних криптопротоколи автентифікації в сучасних засобах встановлення захищеного аудіо- та відео зв’язку, було виявлено загальні недоліки в методах забезпечення стійкої автентифікації в засобах інтернет-телефонії, до них відносяться: покладання на сервіси, що виступають третьою довіреною стороною, в таких методах важко встановити які конфіденційні дані користувач видав про себе та які передав «з рук, в руки», група відомих компаній типу Google, Meta, Microsoft володіють певною інформацією про користувачів, а з використанням біометричних методів автентифікації компанії можуть заволодіти, ще й біометричними даними, більшість проаналізованих застосунків або мають можливий доступ до даних або принаймні зберігають метадані, що потенційно компрометують конфіденційність даних користувачів, також застосунки сильно залежать від централізованих серверів, що також можуть зазнати атак, та номеру телефону користувача, що компрометує анонімність, оскільки дізнавшись номер телефону, що є конфіденційною інформацією, можливо навіть дізнатись, де користувач знаходиться.

Всі ці недоліки є критичними для кібербезпеки користувачів, оскільки можуть становити пряму загрозу деяким аспектам їхнього життя та конфіденційності їх персональних даних, це в свою чергу завдасть неймовірних репутаційних збитків розробникам, так і юридичним та фінансовим, з якими ці

користувачі взаємодіють. Конфіденційні дані користувачів є дуже привабливою здобиччю для злочинців, тому, що ці дані можна продати зацікавленим сторонам, також володіння персональною інформацією користувача можна використати в різноманітних формах соціальної інженерії, що, знову ж таки, може становити пряму загрозу деяким аспектам нормального життя користувача, тому з точки зору безпеки слід очікувати постійних спроб викрадення конфіденційної інформації.

Враховуючи, всі ці вагомі причини, актуальною є розробка безпечного та надійного модулю автентифікації для забезпечення достатнього рівня захисту, що постійно буде готовим до викликів з боку, завжди зацікавлених, зловмисників.

1.5 Висновки з розділу

В результаті проведеного аналізу засобів інтернет-телефонії, було розглянуто програмні засоби встановлення захищеного аудіо- та відео зв’язку, проаналізовано їх функціонал, виявлено індивідуальні недоліки кожного з засобів.

Далі було проведено їх порівняльний аналіз за характеристиками: підтримка апаратних засобів, мультиплатформеність, шифрування, анонімність, збереження даних, інтеграція з іншими сервісами. Після чого було виявлено загальний недоліки розглянутих засобів.

Було здійснено аналіз методів автентифікації сторін. Спочатку було приведено класифікацію методів методів автентифікації сторін, та розділено їх на методи засновані на знаннях, володінні та властивостях, до відповідної категорії було віднесено відомі методи. Далі наведено класифікацію за кількістю сторін, результатом якої стало обґрунтування, щодо того, чому потрібно використати саме двосторонню автентифікацію. Наступним чином було розглянуто поділ протоколів автентифікації на прості, суворі та доведення з нульовим розголошенням знання.

Розглянувши прості протоколи було досягнуто висновку, що доцільно використовувати багатофакторну автентифікацію (MFA), оскільки вона може істотно зменшити імовірність викрадення особистих даних в інтернеті, шляхом використання кількох простих методів, та було наведено приклад формування модулю автентифікації клієнт-сервер, за використання геш-функції SHA-3 та JWT-

токену.

Далі було проведено аналіз криптографічних протоколів автентифікації, в якому наведено аналіз протоколу Фейге-Фіата-Шаміра, що є протоколом доведення з нульовим розголошенням знань, протоколу MS-CHAPv2, що базується на односпрямованій геш-функції та протоколу Otway-Rees, побудований на алгоритмі симетричного шифрування. Після чого, було проведено порівняльний аналіз, в рамках висновків даного аналізу, було визначено як переваги, так і недоліки кожного з них в контексті використання в розробці модулю автентифікації, що надає базис для подальшого вибору криптопротоколу автентифікації.

На основі виконаного аналізу було здійснено постановку задачі дослідження в межах комлексної бакалаврської дипломної роботи. В межах цього етапу базуючись на результатах аналізу засобів інтернет-телефонії, аналізу методів автентифікації сторін та аналізу криптографічних протоколів автентифікації, а саме на виявлених недоліках в забезпечені надійної автентифікації в засобах інтернет-телефонії, сформульовано та обґрунтовано важливість розробки стійкого модулю автентифікації сторін.

2 АРХІТЕКТУРА МОДУЛЯ АВТЕНТИФІКАЦІЇ СТОРІН

2.1 Узагальнена архітектура засобу захищеної інтернет-телефонії

В контексті розгляду інформаційної безпеки програмні застосунки мають певну узагальнену архітектуру, яку потрібно розглянути. Для початку було розглянуто узагальнену архітектуру засобу інтернет-телефонії (рис. 2.1). Цей програмний застосунок містить такі блоки: підмодуль автентифікації користувачів, сервер автентифікації, підмодуль обробки JWT-токенів, підмодуль криптографічного протоколу автентифікації, модуль захищеного зв’язку та сигнальний сервер.

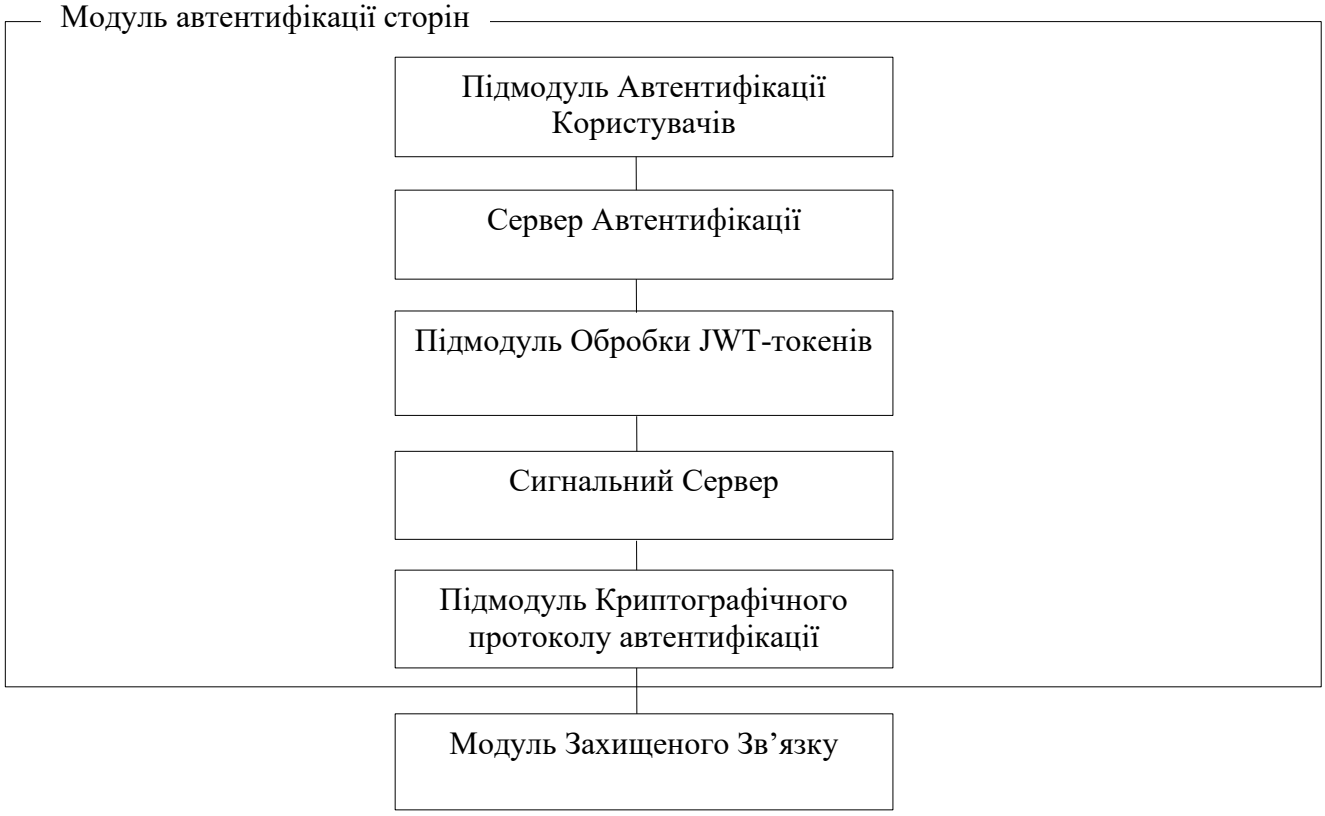


Рисунок 2.1 - Узагальнена архітектура засобу захищеної інтернет-телефонії

Модуль автентифікації сторін, включає в себе декілька підмодулів, що виконуються послідовно. Першим розглядається підмодуль автентифікації користувачів, в цьому підмодулі відбувається автентифікація клієнта – користувача з сервером автентифікації, на вхід надходять дані користувача, ці дані користувача спочатку надходять до серверу автентифікації, в разі успішного першого етапу

автентифікації, шляхом перетворень даних користувача за допомогою геш-функції та порівняння їх з тими, що зберігаються в базі даних. Починається другий етап, а саме формування JWT-токену, на базі деяких даних отриманих в першому етапі та секретних даних серверу автентифікації, далі відбувається взаємодія з сервером, результатом якої є перевірка валідності токену, якщо токен дійсний, користувач вважається автентифікованим перед сервером та отримує доступ до наступного кроку. Наступним в черзі виконання є підмодуль криптографічного протоколу автентифікації, в цьому підмодулі, користувач взаємодіє з іншим користувачем, результатом виконання цього модуля є взаємна автентифікація користувачів один перед одним, за допомогою криптографічного протоколу автентифікації. Далі в разі успішної взаємної автентифікації, йде модуль захищеного зв’язку та сигнального сервера, в якому відбувається власне встановлення захищеного зв'язку і який є іншою частиною комплексної бакалаврської дипломної роботи, тому не розглядається в межах цієї пояснювальної записки.

2.2 Структура модуля автентифікації сторін

Модуль автентифікації сторін складається, в першу чергу, з таких підмодулів: автентифікації користувачів, серверу автентифікації та криптографічного протоколу автентифікації. Підмодуль автентифікації користувачів реалізує збирання даних користувачів, по типу паролю, логіну та ідентифікатора у вигляді електронної пошти, далі відбувається їх перетворення за допомогою геш-функції та надсилання і зберігання в захищеному вигляді на базу даних, також до функції відноситься процес проходження автентифікації, що складається зі збирання даних введених користувачем, що проходить автентифікацію, а саме паролю та логіну, далі відбувається пошук збереженого геш-значення в базі даних за логіном, потім формується геш-значення з введених користувачем даних, сформоване геш-значення порівнюється з тим, що збережене за цим користувачем в базі даних, якщо при порівняні геші збігаються, перший етап автентифікації пройдений. В разі проходження першого етапу автентифікації, модуль створює JWT-токен, він є валідним на протязі години, за допомогою нього

користувач матиме фактичний доступ до ресурсів застосунку, далі відбувається перевірка валідності токену, сервером автентифікації, якщо він валідний користувач має змогу використовувати основний функціонал застосунку. Якщо користувач хоче встановити зв’язок з іншим користувачем, починає працювати блок криптографічного протоколу для здійснення двосторонньої автентифікації за допомогою криптографічного протоколу.

Детальніше структуру модуля автентифікації сторін наведено на рисунку 2.2.

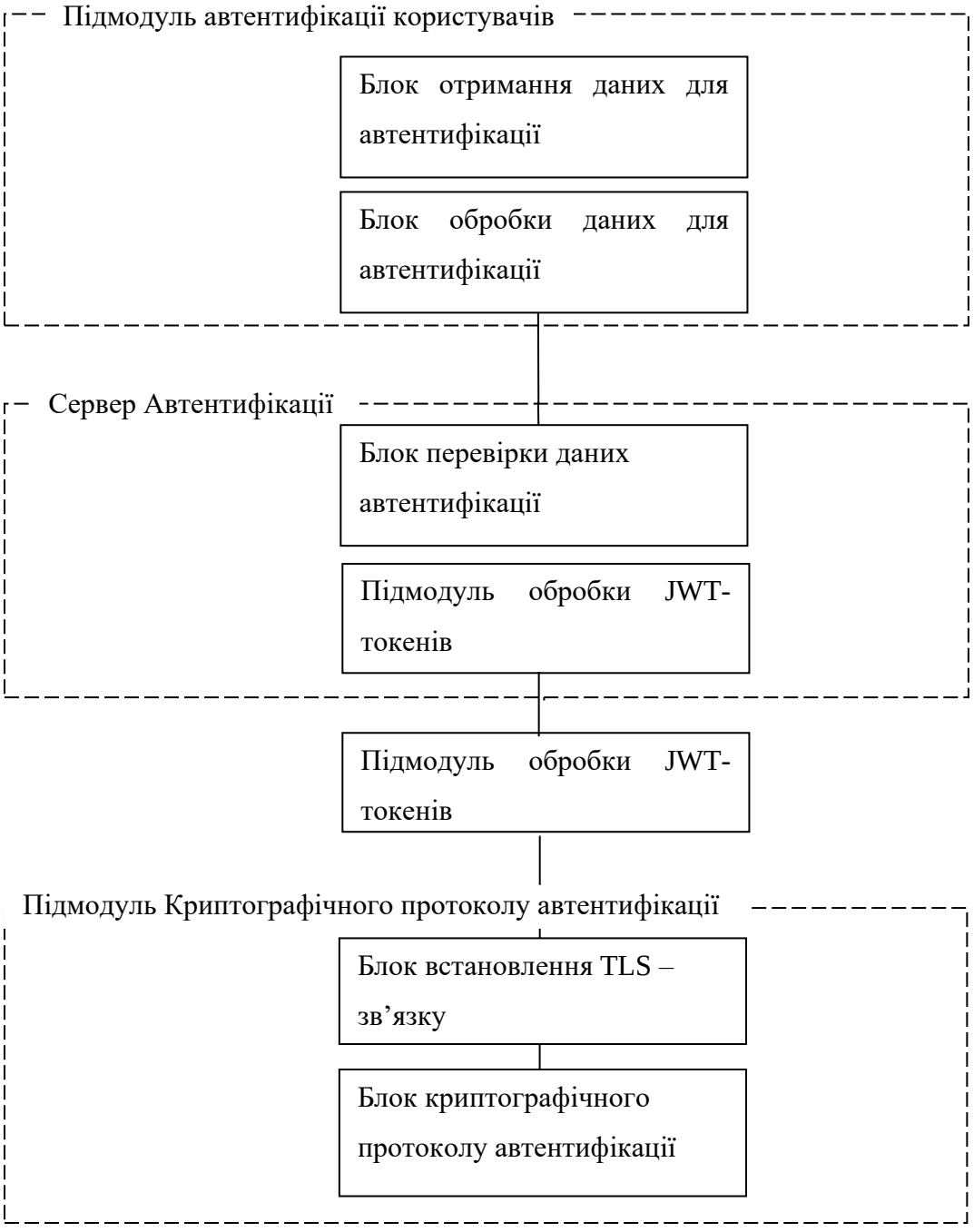


Рисунок 2.2. Структура автентифікації сторін

Отже базуючись на розробленій узагальненій структурі засобу захищеної інтернет-телефонії, було розроблено структуру модуля автентифікації, потрібно обґрунтувати вибір криптографічних алгоритмів.

2.3 Обґрунтування вибору криптографічних алгоритмів

Базуючись на вище проведеному порівняльному аналізі криптографічних протоколів автентифікації, було виявлено недоліки кожного з них, в результаті кожен з протоколів не можна вважати оптимальним, для забезпечення стійкості та швидкодії одночасно, тому варто розглянути та запропонувати власний криптографічний протокол автентифікації, що використовуватиме односпрямовану геш-функцію. Нижче приведено узагальнений опис криптопротоколу двосторонньої автентифікації заснованого на використанні односпямованої геш-функції SHA-3-256. Для опису протоколу будуть використані такі позначення:

- A, B - суб’єкти, сторони, що встановлюють зв’язок;
- I_A, I_B - ідентифікатори сторін A та B;
- R_A, R_B -випадкові числа сторін A та B;
- h - геш-функція SHA-3;
- h_A, h_B - значення геш-функції для сторін A та B відповідно.

Для роботи протоколу необхідне виконання умови, згідно до якої ідентифікатори сторін A та B відомі обом учасникам крипто протоколу. Ініціатором встановлення зв’язку будемо вважати суб’єкта B.

З самого початку сторона B надсилає своє випадкове число стороні A.

$$A \leftarrow B: R_B.$$

Сторона A обчислює геш-значення від ідентифікатора сторони B та випадкових чисел обох сторін, та надсилає стороні B по відкритому каналу геш-значення разом з випадковим числом сторони A.

$$A \rightarrow B: R_A, h_A(I_B, R_A, R_B)$$

Сторона B отримує геш-значення та випадкове число сторони A, та генерує своє геш-значення hB(IB,RA,RB) порівнює його із hA(IB,RA,RB), якщо вони ідентичні, то сторона A автентифіковна, якщо вони не є ідентичними, автентифікація не пройдена.

$$B: h_B(I_B, R_A, R_B) =? h_A(I_B, R_A, R_B)$$

Далі сторона B обчислює геш-значення від ідентифікатора сторони A та випадкових чисел обох сторін, та надсилає стороні A по відкритому каналу геш-значення.

$$A \leftarrow B: h_B(I_A, R_A, R_B)$$

Сторона A обраховує своє геш-значення від випадкових чисел обох сторін та ідентифікаторі сторони A - hA(IA,RA,RB) та порівнює його із отриманим від сторони B, якщо геш-значення рівні – автентифікація пройдена успішно, якщо ж ні – автентифікація провалена.

$$A: h_A(I_A, R_A, R_B) =? h_B(I_A, R_A, R_B)$$

Після опису узагальненого алгоритму криптопротоколу потрібно його проаналізувати використовуючи BAN-логіку [16].

Спочатку було введено початкове значення $X = I_B, R_B, R_A$, після чого було складено ідеалізовану форму $AseesA^{\leftrightarrow}$ x B - стороні надіслано повідомлення, що містить випадкові числа обох сторін та ідентифікатор сторони B.

$$\frac{BbelievesAbelievesX}{BbelievesX}$$

сторона B вірить у те, що A вірить у X, тому сторона B вірить X.

$$AbelievesB\{H(X)\}$$

сторона A вірить у те, що сторона B розділяє геш-значення та порівнює його

із отриманим від сторони А. Залишається застосувати логічні постулати до початкових значень і послідовності тверджень для виявлення довіри частинам протоколу

$$\frac{AbelievesX \rightarrow^{X} BseesH_B(X)=H_A(X)}{BbelievesAsaidX}.$$

Якщо А вірить у те, що В має секретний ключ Х та В створив свій геш-код на основі Х і порівняв із отриманим, то В вірить, що А надіслав Х. З даного аналізу, де було застосовано BAN-логіку, можна зробити такі висновки:

- в остаточному результаті за допомогою цього криптографічного протоколу можна досягти коректної односторонньої автентифікації із наданням доступу або відміні у ньому від сторони В.
- в даному криптографічному протоколі не міститься жодних лишніх кроків, які збільшують роботу протоколу та заважають реалізовувати його точну ціль.
- вхідне повідомлення необхідно передавати після застосування геш-функції, для збереження конфіденційності, в випадку перехоплення повідомлення у відкритому каналі зв’язку.
- в даний криптографічний протокол не потрібно вводити жодного додаткового кроку.

Отже, після аналізу криптографічного алгоритму двосторонньої автентифікації за допомогою випадкових чисел та ключової геш-функції SHA-3, можна дійти висновку, що він є коректним, враховуючи результати аналізу за допомогою BAN-логіки.

2.4 Структура JWT-токену

JSON Web Token - це стандарт токена доступу на основі JSON. Як правило, використовується для передачі даних для автентифікації в клієнт-серверних програмах. В даному випадку та в загальному токени створюються сервером, підписуються секретним ключем і передаються клієнту, який надалі використовує

цей токен для підтвердження своєї особистості.

Токен JWT складається з трьох частин [17]:

– заголовок (header);

– корисне навантаження (payload);

– підпис(verifying siganture).

Перші два елементи – це JSON об'єкти певної структури. Третій елемент обчислюється на основі перших і залежить від обраного алгоритму.

Заголовок - це JSON-об'єкт, який містить метадані про токен, такі як тип токена (зазвичай "JWT") та алгоритм підпису, використовуваний для генерації підпису. В якості алгоритму гешування було обрано алгоритм HMAC-SHA512 [18] (HS512). HS512 базується на геш-функції SHA2-512 та використовує спільний ключ для обчислення підпису. Даний алгоритм має конкурентів у вигляді RS512 [19] та ES512[19] їх порівняння наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльний аналіз проаналізованих засобів

Алгоритм	Складність використання	Ключі	Розмір геш-значення	Підтримка асиметричних ключів	Швидкодія
HS512	Низька	Симетричні	Малий	Ні	Висока
RS512	Висока	Асиметричні	Великий	Так	Низька
ES512	Висока	Асиметричні	Великий	Так	Низька

Порівнюючи даних три алгоритми можна дійти висновку, що HS512 має кращий показник швидкодії та не потребує використання публічного ключа, на відміну від двох інших алгоритмів, а за параметром стійкості не поступається конкурентам.

Отже структура заголовку складатиметься з двох полів, а саме поля типу токену та поля обраного алгоритму:

– “typ”: “JWT”;

– “alg”: “HS512”.

Корисне навантаження (Payload) - це JSON-об'єкт, який містить інформацію,

яку потрібно передати за допомогою токена. Він може містити будь-яку кількість полів з корисною інформацією, наприклад ідентифікатор користувача, ролі, термін дії токена та інші атрибути. Ця частина не є конфіденційною і може бути декодована кожною стороною, яка отримує токен. Нижче приведено поля, що використовуються в корисному навантаженні:

– “Subject” – це поле вказує суб'єкта токена, тобто логін конкретного користувача або сутності, для якої токен видано, що становить саму базову перевірку на те, хто автентифікується за допомогою токену та кому конкретно токен був виданий;
– “Username” – це поле, вказує логін користувача, для якого формується токен;
– “Expiration Time” – це поле вказує час, коли токен стає недійсним, містить в собі повну дату(число, місяць, рік) та час включаючи секунди. В рамках застосунку, час життя токену буде встановлено одну годину, що обмежить час його використання та викличе необхідність у повторній автентифікації, що ускладнить теоретичним злочинцям процес користування чужим обліковим записом;
– “JWT ID” – це унікальний ідентифікатор токена. Він складатиметься з сформованого геш-значення на основі логіну та паролю користувача, використовуючи функцію гешування SHA-3, за рахунок цього можна відстежувати або ідентифікувати конкретний JWT токен.

Підпис (Signature) - це підписана частина токена, яка допомагає перевірити цілісність даних. Для генерації підпису буде використовуватись заголовок та корисне навантаження, а також секретний ключ, який відомий тільки серверу. Підпис дозволяє перевірити, чи були дані в токені змінені після його створення, сам підпис здійснюється за допомогою алгоритму HMAC-SHA512.

В результаті сформований токен, закодований за допомогою стандарту кодування base64 матиме подібний вигляд - "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjoiYWZzIiwiaWF0IjoxNjg2NTk3MDg1LCJleHAiOjE2ODY2MDA2ODUsImp0aSI6ImFmcyJ9.as3jyJTXDhrc8-

5M0XDX5iz_oLdU_tLdbkGzSXjpQ03tzRaZVsfZGMme2QG5WfJowwAM6GzB1slV
EPj3Hd08Hg".

В ньому через кому записано заголовок, корисне навантаження та останньою сама частина підпису, що складається з заголовку, корисного навантаження та секретного ключа. Розроблена структура токену буде далі використана при розробці програмного застосунку.

2.5 Висновки з розділу

В даному розділі було розроблено та проаналізовано узагальнену архітектуру засобу захищеної інтернет-телефонії, загальну архітектуру було розподілено на модуль автентифікації сторін та модуль захищеного зв’язку. Після цього було розроблено модуль автентифікації сторін, що поділяється на певні підмодулі, які реалізують відповідні етапи автентифікації між сервером та користувачем, та між користувачами.

Далі було обґрунтовано використання власного криптографічного протоколу автентифікації. Коректність розробленого протоколу було проаналізована за допомогою BAN-логіки, що формально доводить його коректність. Після чого було розроблено структуру JWT-токену, де детально описано які поля використовуватимуться та яку роль вони відіграють.

В результаті виконання даного розділу було сформовано та розроблено узагальнену структуру засобу захищеної інтернет телефонії, структуру модуля автентифікації сторін, криптографічний алгоритм двосторонньої автентифікації користувачів та структуру JWT-токену, для автентифікації користувачів перед сервером, всі ці розробки стануть базисом для реалізації алгоритмів програмного застосунку, як загальних, так і окремих модулів.

3 АЛГОРИТМИ РОБОТИ МОДУЛЯ АВТЕНТИФІКАЦІЇ СТОРІН

3.1 Узагальнений алгоритм роботи засобу захищеної інтернет-телефонії

Базуючись на розробленій узагальненій структурі захищеного засобу інтернет-телефонії, та її модулів було розроблено узагальнений алгоритм роботи засобу захищеної інтернет-телефонії. Схему алгоритму роботи засобу захищеної інтернет-телефонії наведено на рисунку 3.1.

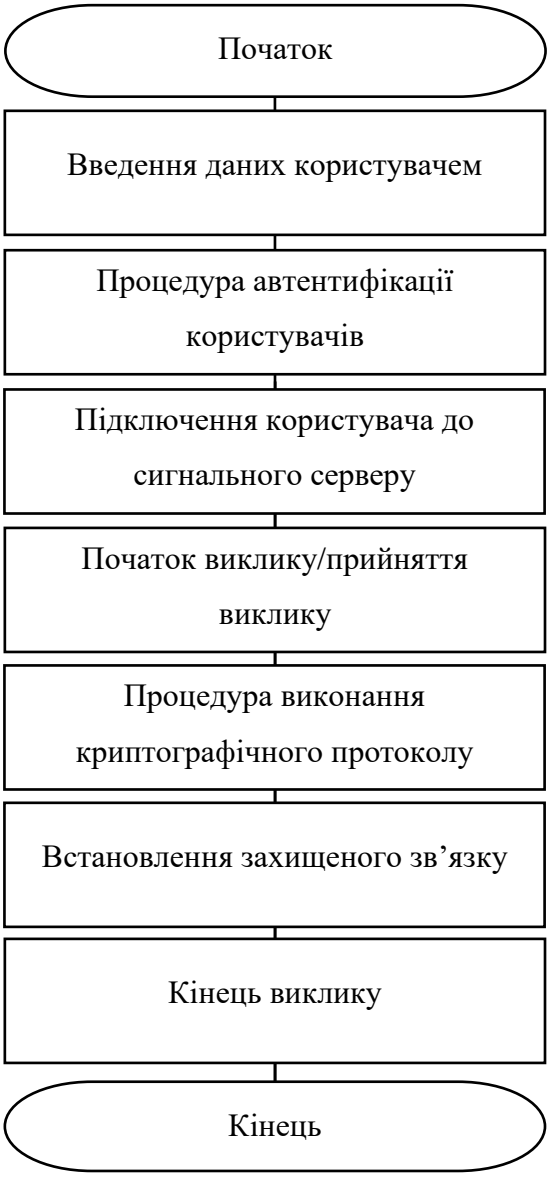


Рисунок 3.1 – Узагальнений алгоритм роботи засобу

Робота засобу починається з введення даних для автентифікації користувачем, далі за отриманими даними відбувається процедура автентифікації користувачів сервером, результатом процедури є проходження двофакторної

автентифікації. Після успішної автентифікації, відбувається підключення до сигнального серверу, за яким слідує виклик одного користувача до іншого, після прийняття виклику починається процедура виконання криптографічного протоколу автентифікації, результатом якої є взаємна автентифікації користувачів один перед одним. Після цього відбувається встановлення захищеного зв’язку, по закінченню зв’язку завершується алгоритм роботи засобу.

3.2 Алгоритм роботи модуля автентифікації сторін

Розроблений алгоритм роботи модуля автентифікації (рис. 3.2) користувачів та його опис представлені нижче.

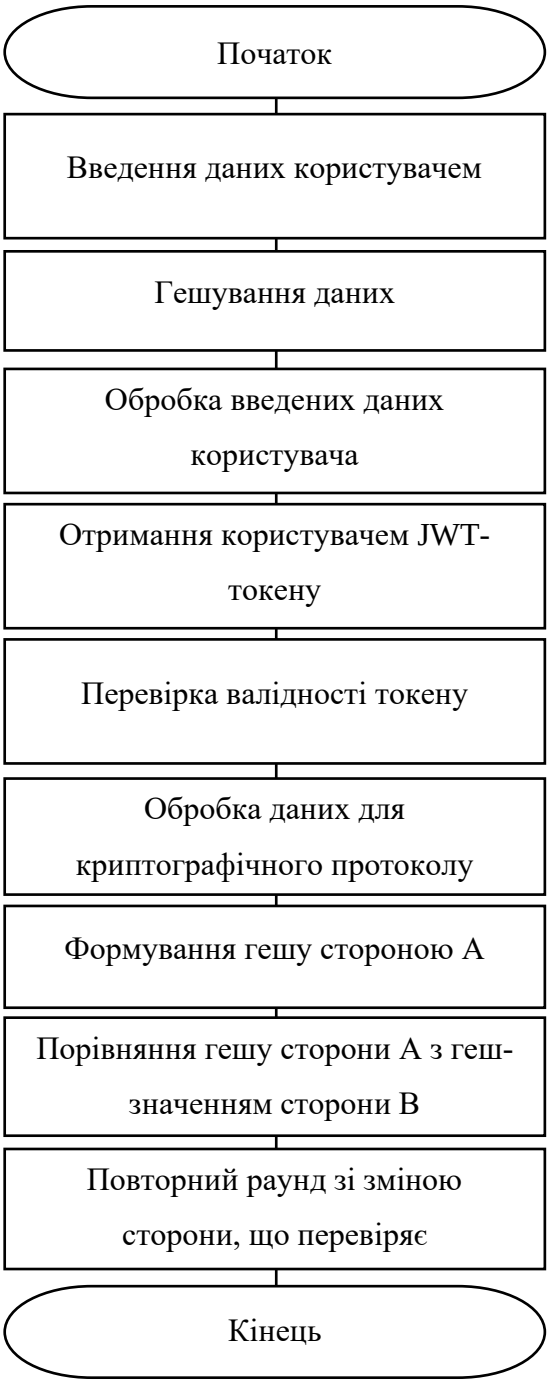


Рисунок 3.2 – Схема узагальненого алгоритму роботи модуля автентифікації

Для початку користувач вводить свої автентифікаційні дані, а саме пароль та логін. За логіном проводиться пошук відповідного логіну паролю, що зберігається в вигляді геш-значення, з введеного паролю користувачем також формується геш-значення, далі ці геш-значення порівнюються, якщо вони ідентичні перший етап автентифікації пройдений і далі на основі логіну та секретного ключа серверу формується JWT-token, із зазначеним строком життя в одну годину. Далі цей токен перевіряється, якщо токен є валідним алгоритм продовжується, після чого користувач має доступ до основної функції застосунку, а саме встановлення зв’язку, тому далі встановлюється TLS зв'язок між двома користувачами, для проведення криптографічного протоколу автентифікації. Спочатку сторона В стороні А надсилає своє випадкове число, після чого сторона А формує геш-значення на основі свого випадкового числа, того числа, що надіслала сторона В, та ідентифікатора сторони В. Потім надсилає сформоване геш-значення стороні В разом зі своїм випадковим числом. Сторона В формує геш-значення на основі таких же значень, що і сторона А, та перевіряє чи геш-значення збігаються, якщо так сторона А автентифікована перед стороною В. Після цього сторона В, формує геш-значення на основі випадкових чисел обох сторін та ідентифікаторі сторони А, надсилає це значення стороні А, сторона А формує геш-значення на основі тих самих значень та перевіряє чи геш-значення збігаються, якщо так сторона В автентифікована перед стороною А.

3.3 Алгоритм автентифікації користувачів

Розробка алгоритму автентифікації користувачів, є не менш важливою задачею, аніж розробка узагальненого алгоритму робити засобу чи криптографічного протоколу, оскільки цей алгоритм є по суті першою лінією безпеки в застосунку. Спочатку користувач на сторінці автентифікації вводить свої автентифікаційні дані, а саме пароль та логін, у відповідні поля логіну та паролю, з цих даних формується JSON-об’єкт, для того, щоб передати його на сервер, на

сервері з цього об’єкту вилучаються дані. Після цього за логіном користувача проводиться пошук відповідного логіну паролю, що зберігається в вигляді геш-значення. Якщо не було знайдено відповідний пароль, це означає, що, або користувач не зареєстрований, або логін введено неправильно. Далі з введеного паролю користувачем формується геш-значення, після чого геш-значення введено паролю та збереженого в базі даних порівнюються, якщо вони ідентичні перший етап автентифікації пройдений і далі на основі логіну, ідентифікаторі, часу життя токену та секретному ключі серверу формується JWT-token, з вказаним строком життя в одну годину. Далі цей токен надсилається до клієнта, клієнт тобто користувач надсилає його на сервер, де токен перевіряється, якщо токен є валідним алгоритм продовжується, після чого користувач має доступ до основної функції застосунку. Даний алгоритм представлено на рисунку 3.3.

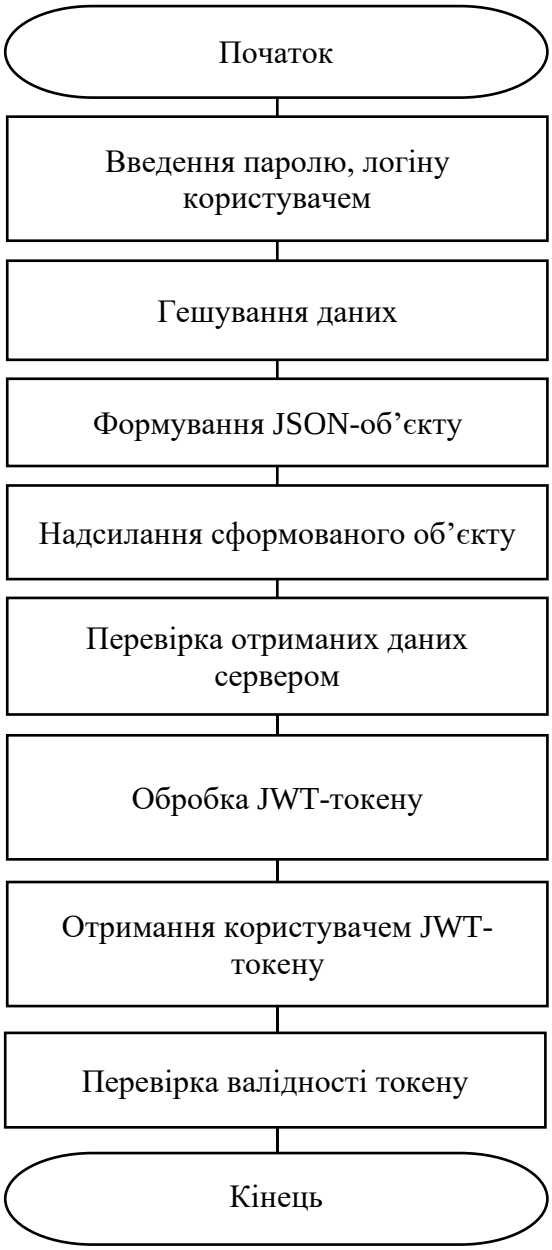


Рисунок 3.3 – Схема алгоритму автентифікації користувачів

Після опису алгоритму автентифікації користувачів варто розробити та описати алгоритм роботи криптографічного протоколу автентифікації.

3.4 Алгоритм роботи криптографічного протоколу автентифікації

Розробка алгоритму роботи криптографічного протоколу автентифікації, одною з найважливіших задач, в контексті розробки захищеного засобу зв'язку. Алгоритм базується на власному криптографічному протоколі, що використовує алгоритм гешування SHA-3 [12] довжиною 256 біт та випадкові числа як криптопримітиви, варто зазначити, що на момент написання роботи алгоритм гешування SHA-3 вважається достатньо стійким для того, щоб протистояти атакам з колізіями та використовувати його у власних розробках.

Спочатку по обидві сторони формуються свої випадкові числа, в Аліси своє число, у Боба своє. Далі Боб надсилає своє випадкове число Алісі, Аліса формує геш-значення на основі ідентифікаторі Боба та випадкових числа Аліси і Боба. Після цього Аліса надчислає геш Бобу разом зі своїм випадковим числом, Боб в свою чергу формує геш-значення на основі ідентифікаторі Боба і випадкових числах Аліси і Боба, порівнює їх, якщо вони ідентичні, то Аліса перед Бобом вважається автентифікованою.

Далі Боб формує геш-значення на основі ідентифікатора Аліси та випадкових числах Аліси та Боба, далі Боб надсилає геш-значення Алісі. Аліса в свою чергу також формує геш-значення на основі ідентифікаторі Аліси та випадкових числах Боба та Аліси, після чого Аліса та порівнює своє геш-значення з тим, що надійшло від Боба, якщо вони ідентичні Боб вважається автентифікованим перед Алісою. На цьому етапі алгоритм роботи криптографічного протоколу автентифікації завершено, а далі починається встановлення захищеного зв'язку для реалізації виклику. Даний алгоритм приведено на рисунку 3.4.

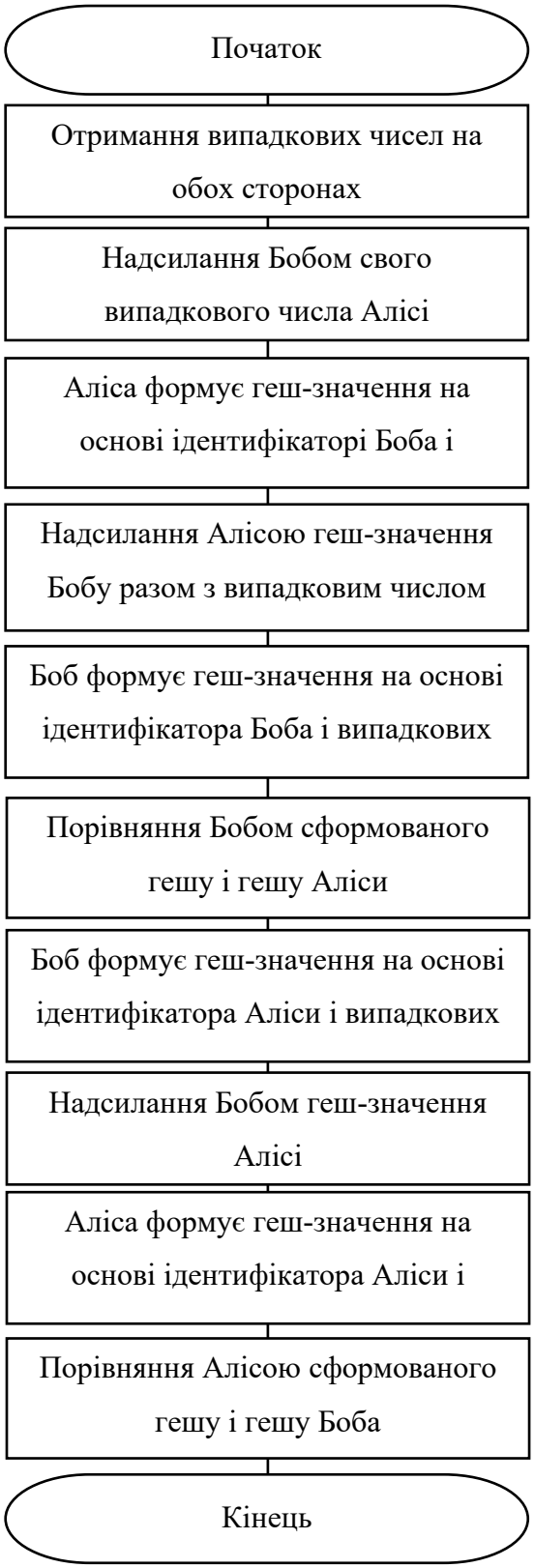


Рисунок 3.4 – алгоритм роботи криптографічного протоколу автентифікації

Після розробки алгоритму роботи криптографічного протоколу автентифікації та інших алгоритмів можна сформувати загальні висновки з розділу.

3.5 Висновки з розділу

В даному розділі було розроблено та проаналізовано узагальнений алгоритм роботи засобу захищеної інтернет-телефонії, узагальнений алгоритм було розподіллено на процедури автентифікації користувачів та криптографічного протоколу автентифікації, що загалом формують модуль автентифікації, алгоритм якого також було розроблено. Після цього стало можливо розробити детальні алгоритми автентифікації користувача з сервером, та користувача з користувачем.

В результаті виконання даного розділу було сформовано та розроблено узагальнений алгоритм роботи засобу захищеної інтернет-телефонії, алгоритм модуля автентифікації сторін, криптографічний алгоритм двосторонньої автентифікації користувачів та алгоритм автентифікації користувачів перед сервером, ці алгоритми допомагають детально розглянути роботу майбутнього програмного засобу, та на етапі їх побудови передбачити певні механізми, недоліки та можливі рішення.

Отже, всі ці розробки враховуючи розроблені - структуру та узагальнений алгоритм засобу захищеної інтернет-телефонії, структуру та алгоритм модуля автентифікації та проведені порівняльні аналізи в першому розділі стануть базисом для вибору засобів розробки, тестування та практичної реалізації програмного застосунку.

4 ТЕСТУВАННЯ ЗАСОБУ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ

4.1 Обґрунтування засобів розробки

Для розробки мобільного застосунку інтернет-телефонії, варто розглянути відповідні для цієї задачі мови програмування, найпоширеніші з яких:

– Swift [20];

– Dart [21];

– JavaScript (JS) [22].

– Kotlin [23];

В контексті розгляду мов програмування в рамках побудови мобільного застосунку, дуже важливим фактором є кросплатформеність, оскільки більшість мобільних пристроїв зазвичай мають операційну систему, або Android, або iOS, тому важливо обрати саме кросплатформенну мову, що забезпечить можливість розробки в першу чергу для цих операційних систем.

Swift є мовою розробки для iOS, а Kotlin головним чином спрямована - для Android. Це означає, що для розробки застосунків для обох платформ, потрібно, або використовувати обидві мови, або одну з кросплатформених мов, до повноцінно ж кросплатформених мов належать JS та Dart . Dart разом з фреймворком Flutter[24] та JS разом з фреймворком ReactNative, дозволяють розробляти кросплатформені застосунки для Android та iOS, що зменшує зусилля та час, необхідний для розробки додатків на обох платформах.

З точки зору екосистеми та засобів розробки, Swift та Kotlin мають потужну екосистему та широкий набір засобів розробки для своїх платформ. Однак, порівняно з Dart та JS, вони можуть бути менш розширеними із точки зору готових компонентів, сторонніх бібліотек та інструментів розробки. Dart та Flutter надають багато готових компонентів та інструментів для розробки інтерфейсу користувача, роботи зі станом додатку, маршрутизації.

Судячи з попереднього опису мов програмування, варто зосередитись саме на мовах JS та Dart.

З точки зору екосистема та ресурсів, React Native має більш широку екосистему, оскільки використовує мову JS, яка вже є популярною та має велику спільноту розробників та багато сторонніх бібліотек. Flutter також має активну спільноту, але його екосистема може бути меншою порівняно з React Native.

Розглядаючи інтерфейс та дизайн, Flutter має власну бібліотеку компонентів Material Design для Android та Cupertino для iOS, що допомагає розробникам легко побудувати красивий та сучасний інтерфейс користувача. React Native використовує компоненти, що відображають стандартні елементи платформи, що може призвести до більш натурального вигляду додатків на кожній платформі.

Незважаючи на деякі зазначені переваги, головним чинником на який потрібно звертати увагу при виборі мови програмування для розробки мобільного застосунку інтернет-телефонії все таки є швидкодія.

JS, який використовується в фреймворку React Native, використовує ядро JS на стороні клієнта, що називається JavaScriptCore. Хоча JavaScriptCore став досить швидким і оптимізованим з часу свого запуску, він все ще має обмеження щодо швидкодії порівняно з виконанням коду природного обчислювальній платформі.

Dart, мова програмування, яку використовує фреймворк Flutter, використовує власний ядро Dart VM (для десктопних платформ) або компіляцію до нативного коду (для мобільних платформ). Dart VM і компіляція до нативного коду дозволяють досягти високої швидкодії виконання. Порівняльний аналіз мов приведено нижче таблиця 4.1.

Таблиця 4.1 – Порівняльний аналіз мов програмування

Характеристика	JavaScript	Dart	Kotlin	Swift
Кросплатформеність	+	+	-	-
Швидкодія	-	+	+	+
Природність	+/-	+/-	+	+

Оскільки для даного засобу важливими показниками якості є кросплатформленість та швидкодія доцільно для реалізації обрати Dart.

4.2 Основні семантичні одиниці програмного коду

Для реалізації програмного застосунку було створено основні класи, які відповідають за інтерфейс користувача, спілкування з сервером, обчислення геш-значень та реалізацію з’єднання для проведення криптографічного протоколу автентифікації.

Для спілкування з сервером який автентифікує користувачів створено файл api-services. Основним поле в даному класі є client в якому записано методи для надсилання запитів REST-API. Також в класі описано два основні методи, а саме login та register, які відсилають запит на сервер за вказаним шляхом, після отримання відповіді надсилають її користувачу. Обидва методи починаються з заголовку, в якому вказано який тип даних буде пересилатися, а саме JSON-об’єкт. Далі формується посилання за яким буде надходити запит, це посилання формується за допомогою бібліотеки Uri, вона є стандартною в мові Dart, використовуючи метод http, параметрами для цього методу є посилання на API та шлях до API, потім створюється змінна response в яку записуються відповідь серверу після обробки запиту.

Файл calculating.services.dart розроблений для обчислення геш-значень паролю користувача, ідентифікатора користувача та значень, що використовуються в криптографічному протоколі. Основним полем є digest, що містить об’єкт класу Digest з бібліотеки pointycastle [25], pointycastle використовується для отримання геш-значень, вона містить певну кількість алгоритмів гешування та алгоритів шифрування. Клас містить два методи calculatingIdent та calculatingHash, перший використовується для обрахування ідентифікатора користувача та повертає дані типу String, другий для обрахування масиву байтів, використовується в криптографічному протоколоі автентифікації користувачів.

Файл double_authorization.services.dart реалізує підключення до сокету і обмін даними між користувачами в межах роботи криптопротоколу автентифікації. Полем класу є socket, ця зміна зберігає дескриптор створеного з’єднання до сокету на сервері.

В класі описані три наступні методи:

– init;

– disconnect;

– sendMessage.

Метод disconnect описує алгоритм, після того як автентифікація за протоколом пройшла успішно, користувачі від'єднуються від сокету після чого зачиняється з'єднання між користувачами.

Наступний метод sendMessage приймає два параметри на вхід, а саме, typeOf - тип події яку потрібно надіслати на сервер та другий параметр data - це дані які потрібно надіслати за допомогою події. Даний метод реалізує відправку події на сервер, разом з даними які потрібно відправити іншому користувачу.

Метод init виконує ініціалізацію зв'язку, тобто даний метод використовується для під'єднання користувача до сокету. Полем є socket – воно приймає два параметри, перший з них це посилання на сокет, а другий це userID, тобто ідентифікатор користувача.

Для спілкування з сервером також були розроблені моделі запитів та відповідей, в яких описуються поля об'єктів які будуть надсилатись у вигляді даних на сервер та користувачу, також описано методи для формування JSON-об'єктів.

Робота додатку починається зі сторінки login_page.dart, для створення інтерфейсу користувача було створено клас LoginPage. Клас містить поля username та password, куди записуються дані з відповідних текстових полів інтерфейсу, що сформовані за допомогою класу TextFormField – цей клас є вбудованим та реалізує поля для введення даних.

Далі потрібно описати метод validate. Даний метод реалізує перевірку правильності введених даних користувачем, для того, щоб користувач не зміг автентифікуватись з пустими полями логіну чи паролю. Якщо валідація даних провалюється, з'являється спливаюче повідомлення, що реалізоване методом _showAlertDialog. Наступним методом в рамках класу є метод _login, він формує модель даних яка відправляє серверу дані введені користувачем.

Наступною є сторінка register_page. Ця сторінка також містить поля

username, password та email, для зберігання введених даних користувачем під час реєстрації в текстові поля інтерфейсу, тут також є метод описаний вище validate та метод _register. Метод _register відповідає за створення моделі даних, в якій передаються всі вказані вище дані, відповідає за гешування паролю користувача та за створення ідентифікатора користувача. В цьому методі створюється об’єкт класу register_request_model в який записуються дані введені користувачем. Після цього модель передається в асинхронну функцію APIService.register(model), яка передає дані серверу, на цьому моменті програма чекає, поки не отримає відповіді від сервера, якщо вона відповідь отримує і відповідь містить дані, то виводиться повідомлення про успішну реєстрацію.

Останньою і головною є сторінка home_page.dart, на якій відбувається криптографічний протокол автентифікації. Містить поля genLocalNumber - дане поле зберігає згенероване локальне секретне число, наступне поле userToAuthenticate – це поле, що зберігає ідентифікатор користувача з яким користувач хоче з’єднатись. Далі поля localIdent, remoteIdent – вони відповідно зберігають ідентифікатори сторони А та сторони В, та поле isAuthorized, що являє собою булівську змінну яка має два стани true – якщо автентифіковано користувача та false – якщо не автентифіковано користувача. Клас _HomePageState реалізує генерацію випадкового числа для криптографічного протоколу двосторонньої автентифікації, включає функцію generateNumber, ця функція генерує число до функції надходить довжина числа якого вимірюється в 32 бітах. Функція authenticate, виконує ініціалізацію автентифікації, функція приймає віддалений ідентифікатор користувача remoteID, формує об’єкт destination який приймає ідентифікатор, тобто куди повинен надійти пакет та об’єкт data, що містить випадкове число, далі функція формує повідомлення з типом ініціалізація автентифікації та відправляє об’єкт data.

Для отримання повідомлення від віддаленого користувача створено метод _eventHandlerInit, який встановлює обробники подій для сокету, є три події initAuth, firstHashMess та secondHashMess, при події initAuth, що являє ініціалізацію, надходить віддалене псевдовипадкове число remoteNumber та

ідентифікатор віддаленого користувача remoteIdent, що розпочав автентифікації, після цього обчислюється геш-значення, що складається з випадкових чисел обох сторін та ідентифікатора, після чого це геш-значення надсилається разом з випадковим числом.

Далі обробляється подія firstHashMes відповідною функцією, до функції надходить об’єкт data, з цього об’єкту вилучається випадкове число, після чого формується геш-значення на основі надісланого випадкового числа, числа та ідентифікатора. Після чого змінній isAuthorized присвоюються результат виклику функції порівняння ListEquality().equals, що порівнює localHash – щойно обрахований геш та remoteHash – надісланий геш. Далі обчислюється нове геш-значення, що зберігається в змінній final secHash на основі remoteNumber – випадкового локального числа, genLocalNumber - випадкового віддаленого числа та ідентифікаторі userToAuthenticate після чого цей геш надсилається.

В наступній події secondHashMess, відбувається отримання надісланого гешу та обчислення свого геш-значення hash на основі свого випадкового числа - genLocalNumber, випадкового віддаленого числа – remoteNumber та свого ідентифікатора widget.localeUser. Після чого геш-значення порівнюються, аналогічного як в попередній події, якщо вони ідентичні автентифікація успішна.

4.3 Блокове тестування модуля автентифікації сторін

Після розробки засобу захищеної інтернет-телефонії, обов’язково потрібно провести блокове тестування засобу, для впевненості в тому, що всі блоки засобу працюють саме так, як очікується.

Для перевірки роботи модуля автентифікації в застосунку потрібно провести тестування автентифікації та реєстрації, тому було розроблено по п’ять тестів, для груп реєстрації та автентифікації. Всі тести для перевірки автентифікації згруповані у групі під назвою Test for Login API рисунку 4.1.

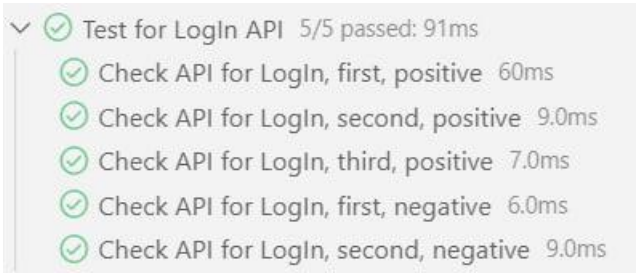


Рисунок 4.1 – Результати тестування групи Test for Login API

Суть проведених позитивних тестів полягає в введені саме коректних даних існуючих користувачів, на що очікується відповідний запит про успішну автентифікацію. Для негативних тестів вводяться неправильні дані користувачів або дані неіснуючих, тобто не зареєстрованих користувачів, що повинно призвести до виникненню помилки, тому результатом таких тестів має бути повідомлення про провал автентифікації через відповідні помилки.

Тести для перевірки процесу реєстрації згруповані у групі під назвою Test for Register API рисунку 4.2.

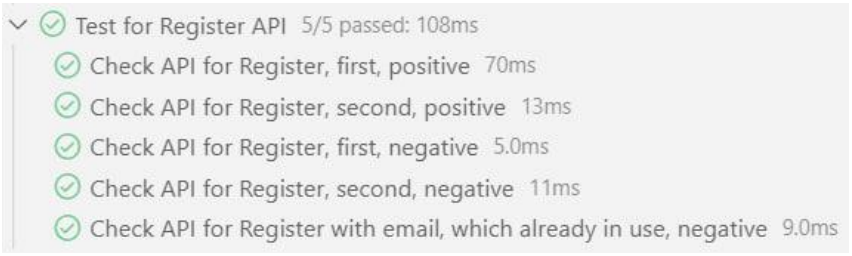


Рисунок 4.2 – Результати тестування групи Test for Register API

Суть проведених позитивних тестів полягає в введені коректних допустимих даних користувачів, на що очікується відповідний запит про успішну реєстрацію. Для негативних тестів вводяться неправильні дані користувачів або деякі дані навмисно не вводяться, наприклад пусті поля логіну чи паролю, що повинно спровокувати виникнення відповідних помилок, тому результатом таких тестів має бути повідомлення про неуспішну реєстрацію через відповідні помилки, що демонструють у спливаючих повідомленнях.

Також важливу роль в роботі всього застосунку відіграє формування геш-значень, як на етапі автентифікації, реєстрації чи криптографічного протоколу,

тому потрібно протестувати коректність формування геш-значень. Всі тести для перевірки геш-значень згруповані у групі під назвою Test for hashing calculating (рис. 4.3).

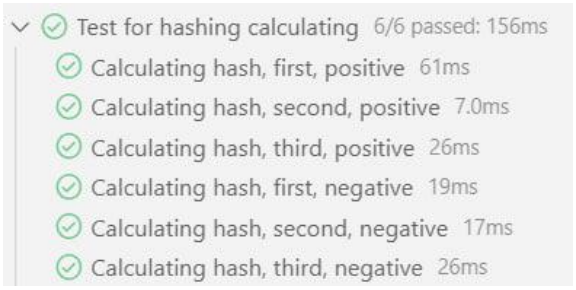


Рисунок 4.3 – Результати тестування групи Test for hashing calculating

Це тестування можна провести за допомогою тестових векторів для гешування з сайту NIST. Суть позитивних тестів полягає в тому, що на вхід подаються такі дані геш для яких вже відомий, якщо геш сформований ідентичний тому який очікується, результатом є успішне гешування. Для негативних тестів, змінюється геш-значення, що очікується на виході, тому в результаті очікується повідомлення про провал.

Результати проведеного блокового тестування показують, що блоки засобу функціонують коректно, це спростить виконання подальшого інтеграційного тестування, оскільки тепер доведено коректну роботу блоків окремо, тому в разі виникнення помилки це заощадить час на пошук помилок, оскільки роботу блоків можна не перевіряти.

4.4 Інтеграційне тестування модуля автентифікації сторін

Після проведеного блокового тестування, потрібно провести інтеграційне тестування для перевірки роботи модулів з точки зору користувача, щоб впевнитись в коректній роботі застосунку.

На початку виконання модуля автентифікації сторін, користувач проходить реєстрацію, користувачу потрібно ввести поштову скриньку, логін та пароль у відповідні поля застосунку рис. 4.4.

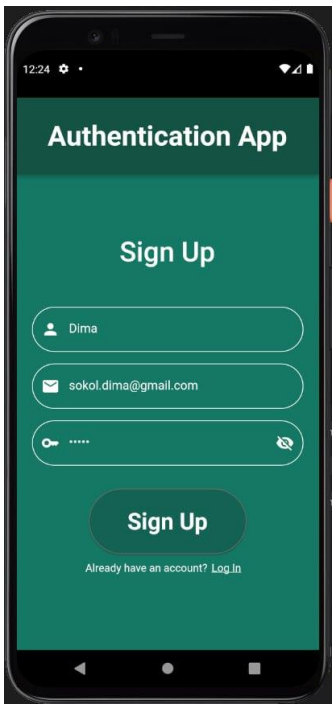


Рисунок 4.4 – Екран проходження реєстрації користувача

Після чого відбувається перевірка введених даних, якщо користувач пропустив якесь з полів, виникне помилка та з’явиться віконце, що показує помилку про пусті поля рис 4.5, після цього відбувається перевірка унікальності логіну, якщо логін є унікальним, виводиться повідомлення про успішну реєстрацію рис. В.1.

Рисунок 4.5– Екран проходження з помилкою про пусті поля

Далі для користувача формується унікальний ID(ідентифікатор), за яким користувача буде ідентифіковано в подальших етапах роботи застосунку, він формується шляхом конкатенації полів логіну та паролю, та формування з них геш-значення за допомогою SHA-3-256, пароль користувача також окремо гешується та в такому вигляді зберігається в базі даних, для використання в подальшій автентифікації. Після реєстрації в повідомленні користувачу пропонується пройти автентифікацію. На етапі автентифікації користувач вводить пароль та логін у відповідні поля в вікні проходження автентифікації рис 4.6. Користувач може натиснути відповідну кнопку, що приховає введений пароль, для уникнення підглядання паролю. Пароль введений користувачем гешується, отримане геш-значення порівнюється зі збереженим для цього користувача геш-значенням паролю, якщо вони не збігаються або користувач ввів неправильний чи пустий логін або пароль, виводиться відповідне повідомлення де зазначено, що введений пароль або логін є неправильними рис. 4.7 або що введено пусті поля 4.8.

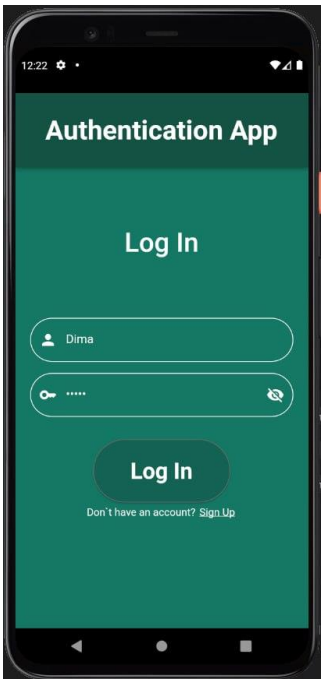


Рисунок 4.6 – Сторінка автентифікації

Рисунок 4.7 – Помилка автентифікації через некоректні дані

Рисунок 4.8 – Помилка автентифікації через пусті поля

Після того як користувач ввів відповідні правильні логін та пароль, формується JWT-токен його строк життя якого є година, після чого потрібно буде отримати новий токен, далі токен надходить користувачу на клієнтську частину програми, після чого цей токен буде перевірено сервером, якщо він є валідним автентифікацію буде завершено, про, що користувача повідомить відповідне повідомлення на екрані рис. 4.9.

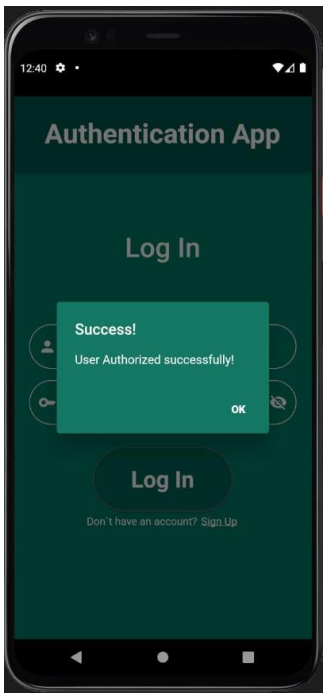


Рисунок 4.9 – Повідомлення про успішну автентифікацію

Інші етапи роботи модуля автентифікації наведено у додатку В. Отже, після проведеного блокового та інтеграційного тестування, необхідно зробити висновки про дієздатність та коректність роботи розробленого модуля для встановлення захищеного зв'язку.

4.5 Висновки з розділу

В цьому розділі було проведено обґрунтування засобів розробки, проаналізовано мови програмування, що дозволяють розробити засіб та його модулі, на результатах аналізу було чітко обґрунтовано вибір мови програмування та фреймворку, за допомогою яких було розроблено модуль автентифікації, результатом обґрунтування стало обрання мови та фреймворку, що в свою чергу забезпечують кращу продуктивність розробки застосунку, оскільки мова є кросплатформенною, а фреймворк допоможе спростити процес розробки, шляхом використання стандартних вбудованих елементів, що спрощує розробку інтерфейсу. Далі було наведено основні семантичні одиниці коду програмного забезпечення, за допомогою яких можна зрозуміти принцип роботи окремих модулів, після цього було проведено блокове автоматизоване тестування, що дало

змогу ефективно вирішувати проблеми, оскільки помилки було чітко видно в межах кожного блоку, що звужує область пошуку моменту де виникала помилка. Після цього було проведено інтеграційне тестування, результатом якого стало встановлення того факту, що і на стороні користувача все працює коректно.

В результаті проведеного тестування засобу захищеної інтернет-телефонії, було виявлено та усунуто помилки, що заважали коректній роботі застосунку, було проведено блокове та інтеграційне тестування, кінцевими результатами яких стала відсутність помилок, що дозволяє дійти позитивного висновку щодо функціонування застосунку.

ВИСНОВКИ

В результаті аналізу поширених засобів інтернет-телефонії було виявлено загальний недолік – він полягає у використанні даними засобами критично конфіденційних даних користувачів, таких як унікальні біометричні дані, які неможливо змінити та номери телефонів, під час автентифікації та в подальшій роботі застосунків. В процесі аналізу відомих методів автентифікації було класифіковано методи автентифікації засновані на: знаннях, володінні та особливостях, також класифіковано методи за сторонами, що приймають участь та факторами, були приведені різні методи автентифікації відповідно до класифікації. Після проведеного аналізу було досягнуто висновків, що для засобів інтернет-телефонії доцільно використовувати двосторонню автентифікацію користувачів, оскільки обидва користувачі будуть впевнені в тому, що вони спілкуються з тою людиною, за яку вона себе видає, а за факторами найкращим методом є саме багатофакторна автентифікація, оскільки потребує представлення декількох факторів для успішної автентифікації.

Проаналізувавши відомі криптографічні протоколи автентифікації було класифіковано їх на ті, що побудовані на основі: симетричних алгоритмів шифрування, асиметричних алгоритмів шифрування, електронного цифрового підпису та на основі односпрямованої геш-функції та розглянуто по одному поширеному протоколу для кожного виду. В результаті аналізу було досягнуто висновку, що представлені відомі криптографічні протоколи не підходять для побудування модулю автентифікації саме в засобі захищеної інтернет телефонії, оскільки деякі з них вимагають значних обчислень на пристрої користувача, деякі не підтримуються стандартами, мають поганий показник швидкодії та потребують складної реалізації чи впровадження додаткових заходів безпеки, загальним недоліком є велика кількість ітерацій обміну даними, що призведе до повільного виконання протоколу мережею інтернет, тому виникла нагальна потреба у побудові власного швидкого протоколу.

Базуючись на результатах аналізу розроблено узагальнену архітектуру

засобу захищеної інтернет-телефонії, структуру модулю автентифікації. Після чого було обґрунтовано вибір геш-функції та представлено розроблений криптографічний протокол, що побудований на основі односпрямованої геш-функції та використанні псевдовипадкових чисел та розроблено структуру JWT-токену. Оскільки аналіз показав потребу у розробці власного методу автентифікації сторін, було розроблено цей метод та доведено його коректність засобами формального аналізу BAN-логіки.

Визначена архітектура засобу дозволила розробити узагальнений алгоритм його роботи та алгоритми роботи його основних складових. Розробка алгоритмів дозволила перейти до їх реалізації за допомогою програмування.

Перед початком розробки було обґрунтовано вибір засобів розробки, після чого було розроблено програмну реалізацію засобу та наведено опис її основним семантичних одиниць. Для доведення коректності прийнятих технічних рішень було проведено тестування засобу.

Основними перевагами розробленого засобу, над проаналізованими відомими аналогами, є відсутність прив’язки до номеру телефону, в засобі не використовуються конфіденційні дані користувачів, водночас реалізується надійна автентифікація користувачів, включаючи двосторонню автентифікацію віддалених користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Cohen R. A random woman joined the wrong Zoom meeting while it was live on The BBC. Insider. URL: https://www.insider.com/woman-joined-wrong-zoom-meeting-live-on-bbc-news-2022-6 (accessed: 14.06.2023).

2 WhatsApp. WhatsApp.com. URL: https://www.whatsapp.com/?lang=en (accessed: 14.06.2023).

3 Specifications >> The X3DH Key Agreement Protocol. Signal Messenger. URL: https://signal.org/docs/specifications/x3dh/ (accessed: 14.06.2023).

4 Skype | Stay connected with free video calls worldwide. Skype | Stay connected with free video calls worldwide. URL: https://www.skype.com/en/ (accessed: 15.06.2023).

5 Transport Layer Security protocol. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/windows-server/security/tls/transport-layer-security-protocol (accessed: 14.06.2023).

6 Documentation. Signal Messenger. URL: https://www.signal.org/docs/ (accessed: 14.06.2023).

7 Specifications >> The Double Ratchet Algorithm. Signal Messenger. URL: https://signal.org/docs/specifications/doubleratchet/ (accessed: 14.06.2023);

8 Daemon J., Rijmen V. The Design of Rijndael: The Advanced Encryption Standard. 2nd ed. Springer, 2020. 300 p;

9 OpenID Connect (OIDC) on the Microsoft identity platform - Microsoft Entra. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/azure/active-directory/develop/v2-protocols-oidc (accessed: 14.06.2023).

10 Christof Paar, Jan Pelzl. SHA-3 and The Hash Function Keccak 21 p. URL: https://www.cryptotextbook.com/download/Understanding-Cryptography-Keccak.pdf (accessed: 14.06.2023).

11 ISO/IEC 9798-4:1999. ISO. URL: https://www.iso.org/en/contents/data/standard/03/14/31488.html (accessed:

14.06.2023).

12 RFC 2759: Microsoft PPP CHAP Extensions, Version 2. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc2759 (accessed: 14.06.2023).

13 RFC 1320: The MD4 Message-Digest Algorithm. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc1320 (accessed: 14.06.2023).

14 Katz J., Lindell Y. Introduction to Modern Cryptography. Chapman and Hall/CRC, 2014. URL: https://doi.org/10.1201/b17668.

15 Anderson R. Security engineering: A guide to building dependable distributed systems. 2nd ed. Indianapolis, IN : Wiley Technology Pub., 2008.

16 A Logic of Authentication. CMU School of Computer Science. URL: http://www.cs.cmu.edu/~dga/15-712/F07/papers/Burrows90.pdf, 1990. (accessed: 18.06.2023).

17 JWT.IO - JSON Web Tokens Introduction. JSON Web Tokens - jwt.io. URL: https://jwt.io/introduction#:~:text=What%20is%20JSON%20Web%20Token, because%20it%20is%20digitally%20signed (accessed: 14.06.2023).

18 RFC 2104: HMAC: Keyed-Hashing for Message Authentication. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc2104 (accessed: 14.06.2023).

19 RFC 7518: JSON Web Algorithms (JWA). IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc7518 (accessed: 18.06.2023).

20 Swift.org. Swift.org. URL: https://www.swift.org/about/ (accessed: 15.06.2023).

21 Dart programming language. Dart programming language | Dart. URL: https://dart.dev/ (accessed: 15.06.2023).

22 JavaScript language overview - JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_overview (accessed: 15.06.2023).

23 Kotlin Programming Language. Kotlin. URL: https://kotlinlang.org/ (accessed: 15.06.2023).

24 Flutter - Build apps for any screen. Flutter - Build apps for any screen. URL:

https://flutter.dev/ (accessed: 15.06.2023).

25 Pointycastle | Dart Package. Dart packages. URL: https://pub.dev/packages/pointycastle (date of access: 18.06.2023).

ДОДАТКИ

ТЕКСТ ЗАСТОСУНКУ. МОДУЛЬ КЛІЄНТА

api_services.dart

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/login_request_model.dart';
import '../models/login_response_model.dart';
import '../models/register_request_model.dart';
import '../models/register_response_model.dart';

import '../config.dart';

class APIService {
  static var client = http.Client();

  static Future<LoginResponseModel> login(
    LoginRequestModel model,
  ) async {
    Map<String, String> requestHeaders = {
      'Content-Type': 'application/json',
    };

    var url = Uri.http(
      Config.apiURL,
      Config.loginAPI,
    );

    var response = await client.post(
      url,
      headers: requestHeaders,
      body: jsonEncode(model.toJson()),
    );

    return loginResponseJson(response.body);
  }

  static Future<RegisterResponseModel> register(
    RegisterRequestModel model,
  ) async {
    Map<String, String> requestHeaders = {
      'Content-Type': 'application/json',
    };

    var url = Uri.http(
      Config.apiURL,
      Config.registerAPI,
    );

    var response = await client.post(
      url,
      headers: requestHeaders,
      body: jsonEncode(model.toJson()),
    );

    return registerResponseJson(
      response.body,
    );
  }
}
```

calculating.services.dart

```
import 'dart:convert';
import 'dart:typed_data';
import 'package:pointycastle/pointycastle.dart';

class Calculating {
  final digest = Digest("SHA3-256");

  Calculating._();
  static final instance = Calculating._();

  String calculatingIdent(strForHash) {
    final hash = digest.process(Uint8List.fromList(utf8.encode(strForHash)));
    return String.fromCharCodes(hash);
  }

  Uint8List calculatingHash(strForHash) {
    return digest.process(Uint8List.fromList(utf8.encode(strForHash)));
  }
}
```

double_authorization.services.dart

```
import 'dart:developer';
import 'package:socket_io_client/socket_io_client.dart';

class DoubleAuth {
  Socket? socket;

  DoubleAuth._();
  static final instance = DoubleAuth._();

  init({required String socketURL, required String? userID}) {
    socket = io(socketURL, <String, dynamic>{
      "transports": ['websocket'],
      "query": {"userID": userID}
    });

    socket!.onConnect((data) {
      log("Socket connected!!");
    });

    socket!.onConnectError((data) {
      log("Connect Error $data");
    });

    socket!.connect();
  }

  sendMessage({required String typeOf, required dynamic data}) {
    socket!.emit(typeOf, data);
  }

  disconnect() {
    socket!.close();
  }
}
```

home_page.dart

```
import 'dart:math';
import 'dart:typed_data';
import 'package:flutter/material.dart';
import 'package:program/services/double_authorization.services.dart';
import 'package:program/services/calculating.services.dart';
import 'package:collection/collection.dart';

class HomePage extends StatefulWidget {
  final String localeUser;

  const HomePage({super.key, required this.localeUser});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final remoteCallerIdTextEditingController = TextEditingController();
  int genLocalNumber = 0;
  String userToAuthenticate = "";
  String remoteIdent = "", localIdent = "";
  Uint8List localHash = Uint8List(0), remoteHash = Uint8List(0);
  bool isAuthorized = false;

  @override
  void initState() {
    super.initState();
    genLocalNumber = generateNumber(32);
    _eventHandlerInit();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Scaffold(
        backgroundColor: const Color(0xFF177965),
        appBar: AppBar(
          toolbarHeight: 100,
          centerTitle: true,
          title: const Text("Authentication App"),
          backgroundColor: const Color.fromARGB(255, 17, 82, 69),
          titleTextStyle: const TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
            fontSize: 32,
            fontFamily: "Roboto"
          ),
          leading: IconButton(
            icon: const Icon(Icons.arrow_back, color: Colors.white,),
            onPressed: () {
              DoubleAuth.instance.disconnect();
              Navigator.pushNamedAndRemoveUntil(context, "/login", (route) => false);
            },
          ),
        ),
        body: _homeUI(context)
      ),
    );
  }

  Widget _homeUI(BuildContext context) {
    return SafeArea(
      child: Stack(
        children: [
          Center(
            child: SizedBox(
              width: MediaQuery.of(context).size.width * 0.9,
              child: Column(
```

```
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      const SizedBox(height: 10),
                      TextField(
                        controller: remoteCallerIdTextEditingController,
                        textAlign: TextAlign.center,
                        decoration: InputDecoration(
                          counterText: "",
                          hintText: "Remote userID",
                          alignLabelWithHint: true,
                          border: OutlineInputBorder(
                            borderRadius: BorderRadius.circular(10.0),
                          ),
                          hintStyle: const TextStyle(
                            color: Colors.white60
                          ),
                        ),
                        style: const TextStyle(
                          fontSize: 28,
                          fontWeight: FontWeight.bold,
                          color: Colors.white
                        ),
                      ),
                      const SizedBox(height: 20),
                      ElevatedButton(
                        style: ButtonStyle(
                          backgroundColor:                                    const
MaterialStatePropertyAll<Color>(Color.fromARGB(255, 20, 98, 83)),
                          padding:                                            const
MaterialStatePropertyAll<EdgeInsets>(EdgeInsets.only(
                              top: 25,
                              right: 50,
                              bottom: 25,
                              left: 50
                            )),
                            shape:
MaterialStatePropertyAll<RoundedRectangleBorder>(RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(100.0),
                              side: const BorderSide(color: Color(0xFF6a6a6a))
                            ))
                        ),
                        onPressed: () {
                          _authenticate(remoteCallerIdTextEditingController.text);
                        },
                        child: const Text(
                          "Authenticate",
                          style: TextStyle(
                            color: Colors.white,
                            fontSize: 30,
                            fontWeight: FontWeight.bold
                          ),
                        )
                      ),
                    ],
                  ),
                ),
              ),
            ),
            if(isAuthorized)
              AlertDialog(
                title:  const Text(
                  "Success",
                  style:  TextStyle(
                    color: Colors.white
                  ),
                ),
                backgroundColor: const Color(0xFF177965),
                content: const Text(
                  "Users authorized",
                  style: TextStyle(
                    color: Colors.white
                  ),
```

```
                    ),
                    actions: <Widget>[
                      TextButton(
                        child: const Text(
                          "OK",
                          style: TextStyle(
                            color: Colors.white,
                            fontWeight: FontWeight.w900
                          ),
                        ),
                        onPressed: () {
                          remoteCallerIdTextEditingController.clear();
                          setState(() {
                            isAuthorized = false;
                          });
                        },
                      )
                    ],
                  )
              ],
            ),
          );
        }

        _eventHandlerInit() {
          int remoteNumber = 0;

          DoubleAuth.instance.socket!.on("initAuth", (data) {
            remoteNumber = data["data"];
            remoteIdent = Calculating.instance.calculatingIdent(remoteNumber.toString() +
widget.localeUser);
            final hash = Calculating.instance.calculatingHash(genLocalNumber.toString() +
remoteNumber.toString() + remoteIdent);
            DoubleAuth.instance.sendMessage(typeOf: "firstHashMess", data: {"hash": hash,
"randomNum": genLocalNumber});
          });

          DoubleAuth.instance.socket!.on("firstHashMess", (data) {
            remoteNumber = data["data"]["randomNum"];
            final hash = Calculating.instance.calculatingHash(remoteNumber.toString() +
genLocalNumber.toString() + localIdent);
            setState(() {
              remoteHash = data["data"]["hash"];
              if(localHash.isEmpty) {
                localHash = hash;
              }
            });
            setState(() {
              isAuthorized = const ListEquality().equals(localHash, remoteHash);
            });
            final secHash = Calculating.instance.calculatingHash(remoteNumber.toString() +
genLocalNumber.toString() + userToAuthenticate);
            DoubleAuth.instance.sendMessage(typeOf: "secondHashMess", data: secHash);
          });

          DoubleAuth.instance.socket!.on("secondHashMess", (data) {
            final hash = Calculating.instance.calculatingHash(genLocalNumber.toString() +
remoteNumber.toString() + widget.localeUser);
            setState(() {
              remoteHash = data["data"];
              if(localHash.isEmpty) {
                localHash = hash;
              }
            });
            setState(() {
              isAuthorized = const ListEquality().equals(localHash, remoteHash);
            });
          });
        }

        _authenticate(String remoteID) {
```

```
            dynamic data = {
              "destination": remoteID,
              "data": genLocalNumber
            };

            userToAuthenticate = remoteID;
            localIdent  =  Calculating.instance.calculatingIdent(genLocalNumber.toString()  +
remoteID);

            DoubleAuth.instance.sendMessage(typeOf: "initAuth", data: data);
          }

          int generateNumber(num bitLength) {
            int min = 1;
            int max = 1;

            for(int i = 0; i < bitLength; i++) {
              max *= 2;
              if(i < bitLength - 1) {
                min *= 2;
              }
            }

            int result = min + Random().nextInt(max - min);
            return result;
          }
        }
```

СТРУКТУРА ПРОЕКТУ

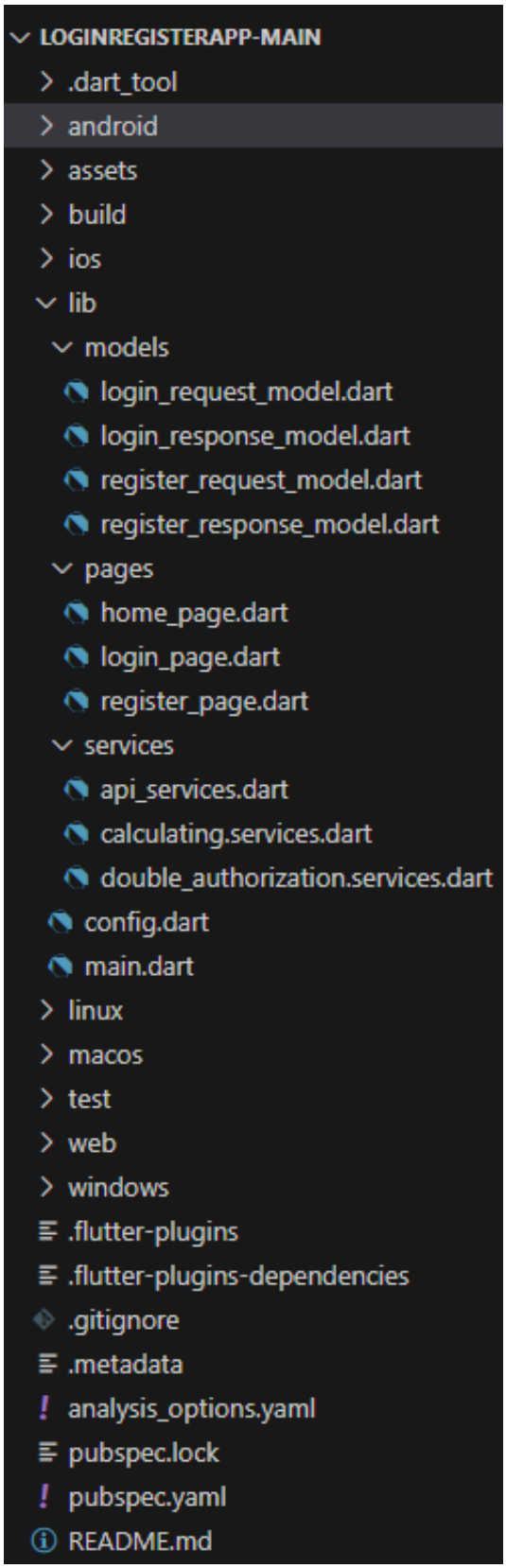


Рисунок Б.1 – Структура клієнтської частини застосунку

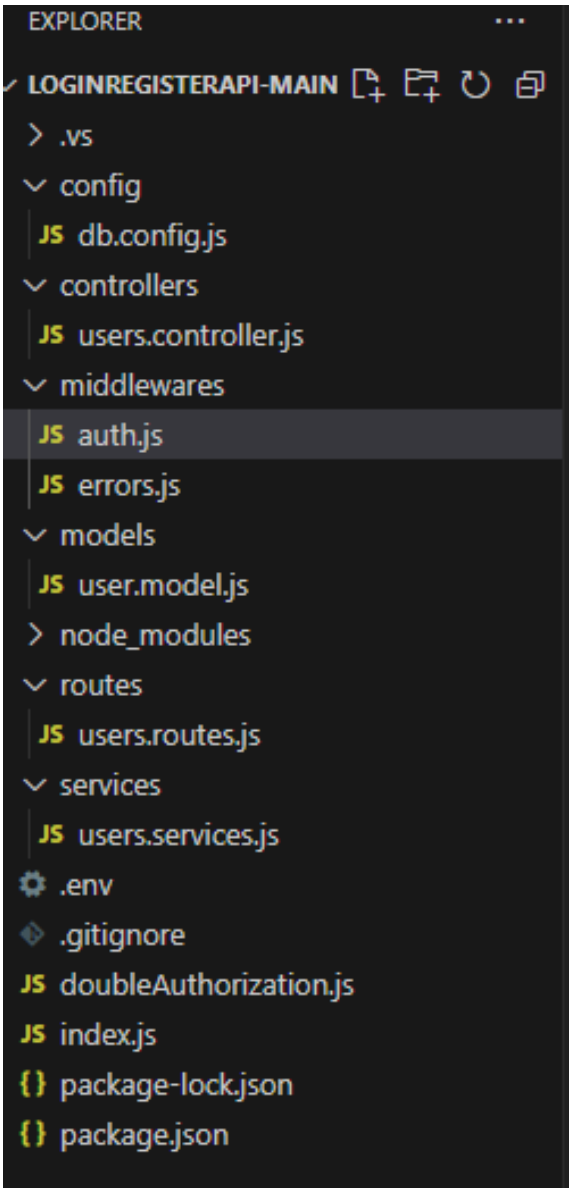


Рисунок Б.2 – Структура серверної частини застосунку

РЕЗУЛЬТАТИ ІНТЕГРАЦІЙНОГО ТЕСТУВАННЯ ЗАСТОСУНКУ

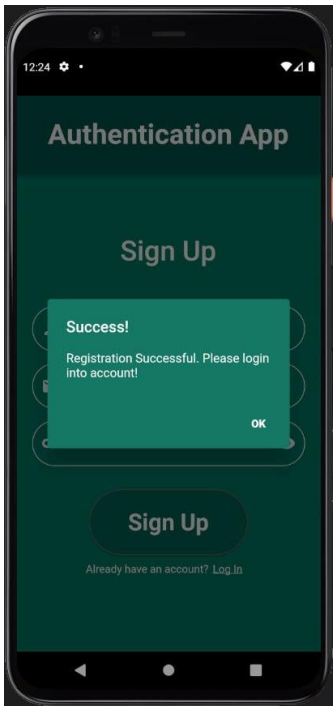


Рисунок В.1 – Повідомлення про успішну реєстрацію

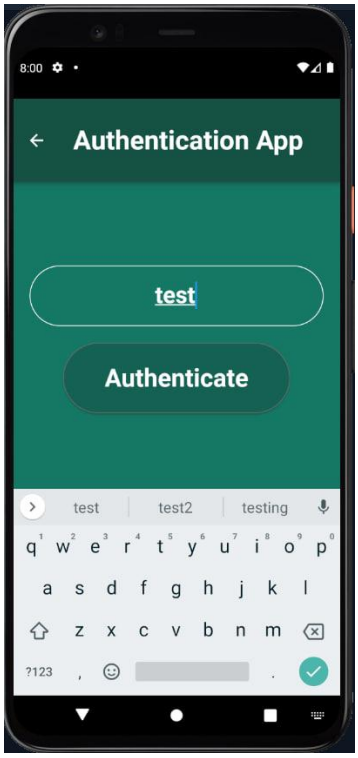


Рисунок В.2 – Форма введення ідентифікатора користувача для взаємної автентифікації

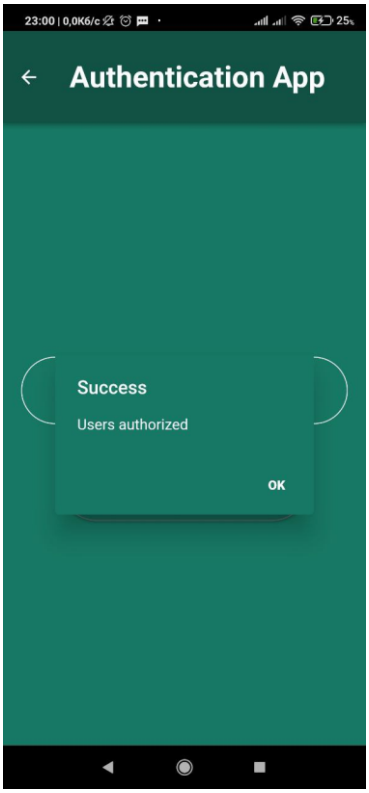


Рисунок В.3 – Спливаюче повідомлення про успішну взаємну автентифікацію на першому пристрої

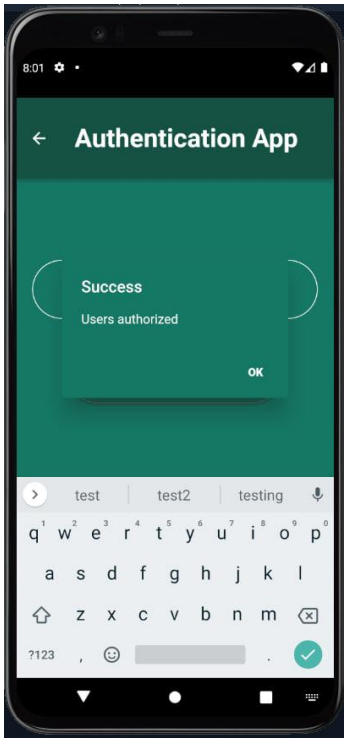


Рисунок В.4 – Спливаюче повідомлення про успішну взаємну автентифікацію на другому пристрої

ПРОТОКОЛ ПЕРЕВІРКИ
БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Засіб захищеної інтернет-телефонії. Частина 1. Модуль автентифікації сторін

Автор роботи: Сокол Дмитро Анатолійович

Тип роботи: бакалаврська дипломна робота.

Підрозділ кафедра захисту інформації ФІТКІ.

Показники звіту подібності Unicheck

Оригінальність – 96,1%. Схожість – 3,9%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Каплун В. А. (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Сокол Д. А. (прізвище, ініціали)

Керівник роботи _____ (підпис) Бакалов Ю. В. (прізвище, ініціали)

ІЛЮСТРАТИВНА ЧАСТИНА

ЗАСІБ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ. ЧАСТИНА 1. МОДУЛЬ АВТЕНТИФІКАЦІЇ СТОРІН

(Назва комплексної комплексної бакалаврської дипломної роботи)

Виконав: студент 4 курсу групи 1БС-19б
Спеціальності 125 Кібербезпека
_____ Дмитро СОКОЛ

Керівник: к. т. н., доцент, доцент каф ЗІ
_____ Юрій БАРИШЕВ
« 16 » червня 2023 р.

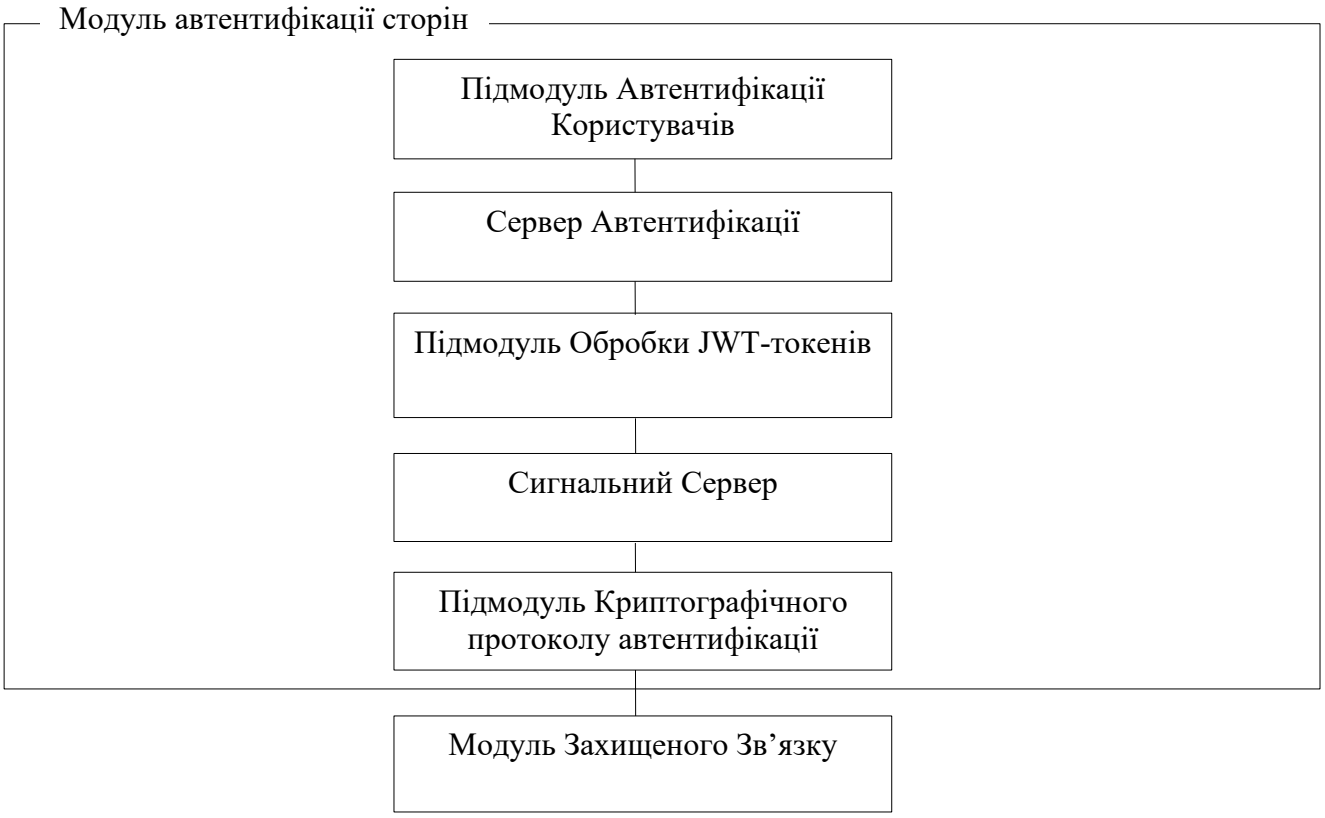
ПОРІВНЯЛЬНИЙ АНАЛІЗ ЗАСОБІВ ІНТЕРНЕТ-ТЕЛЕФОНІЇ

Характеристика	Signal	WhatsApp	Skype
Підтримка апаратних засобів	Немає	Немає	Фізичний ключ YubiKey
Мультиплатформеність(Операційні Системи)	iOS, Android, Microsoft Windows, Linux, macOS	Android, iOS, Microsoft Windows macOS, S40, Tizen, KaiOS	Microsoft Windows, macOS, Android, iOS, Symbian OS, Windows Phone, Linux, BlackBerry OS, webOS
Шифрування	Шифрування на стороні користувача, застосовується E2E шифрування	Шифрування на стороні користувача, застосовується E2E шифрування	Шифрування на стороні користувача але E2E шифрування не застосовується для всіх типів повідомлень
Анонімність	Анонімний	Не анонімний, потрібен номер телефону	Не анонімний, потрібен, обліковий запис Microsoft
Збереження даних	Зберігаються на серверах, після доставки повідомлення вони видаляються з сервера, не зберігає метадані	Зберігаються на серверах, можливе видалення повідомлень з обох сторін	Зберігаються на серверах, можливе видалення повідомлень з обох сторін
Інтеграція з іншими сервісами	Обмін файлами, можливість інтеграції з деякими сервісами	Обмін файлами інтеграція з різними сервісами, такими як iCloud	Обмін файлами, інтеграція з різними сервісами, такими як OneDrive

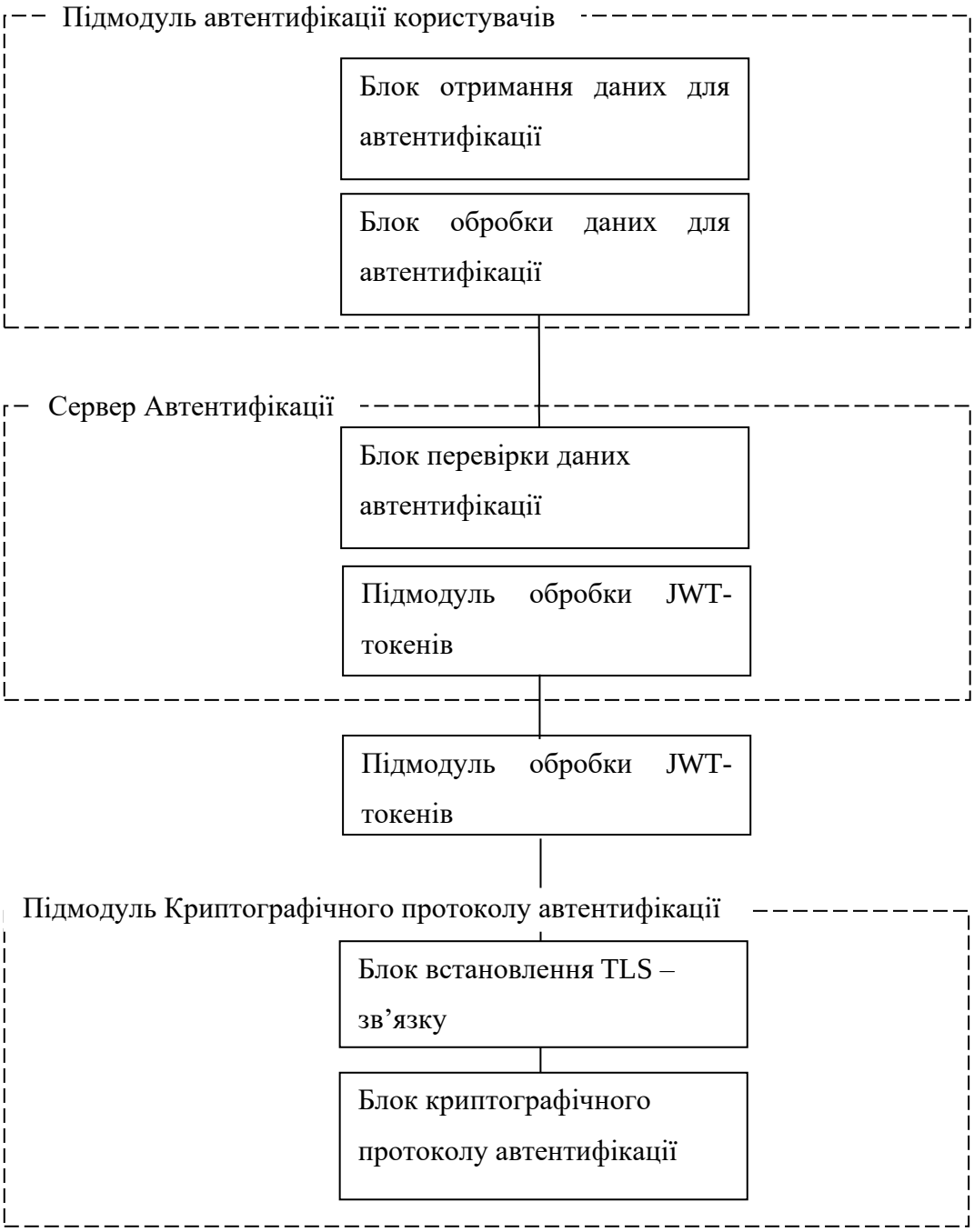
АНАЛІЗ КРИПТОГРАФІЧНИХ ПРОТОКОЛІВ АВТЕНТИФІКАЦІЇ

Характеристика	Протокол Фейге—Фіата—Шаміра	Протокол MS-CHAPv2	Протокол Otway-Rees
Стійкість	Висока	Середня	Середня
Швидкодія	Низька	Помірна	Висока
Ефективність в використанні ресурсів	Середня	Висока	Середня
Простота реалізації	Низька	Середня	Висока
Підтримка стандартами	Ні	Так	Ні
Масштабованість	Середня	Висока	Помірна

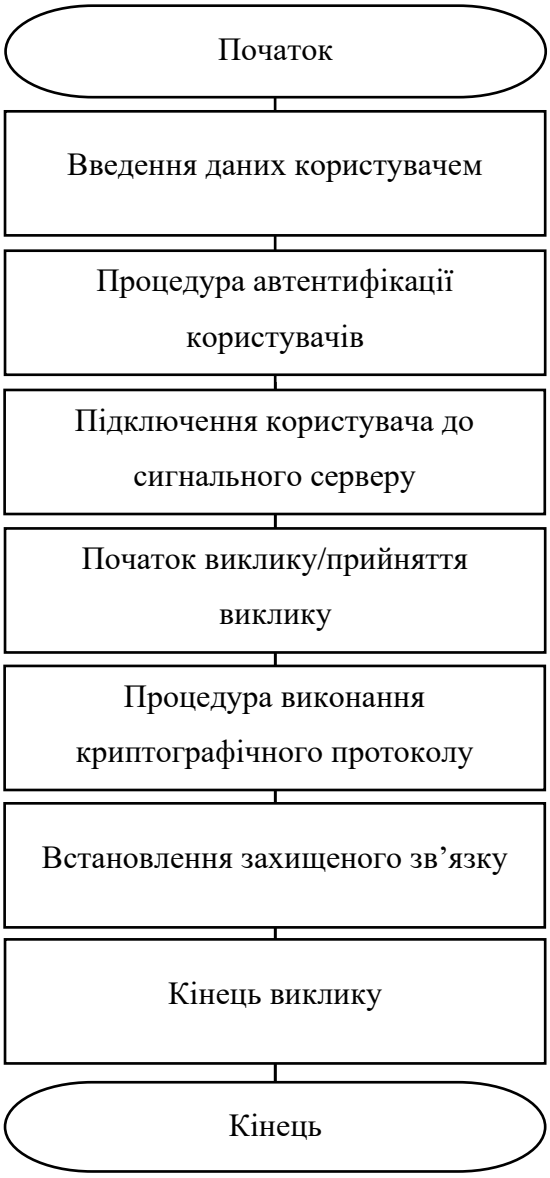
АРХІТЕКТУРА ЗАСОБУ ЗАХИЩЕНОЇ ІНТЕРНЕТ-ТЕЛЕФОНІЇ



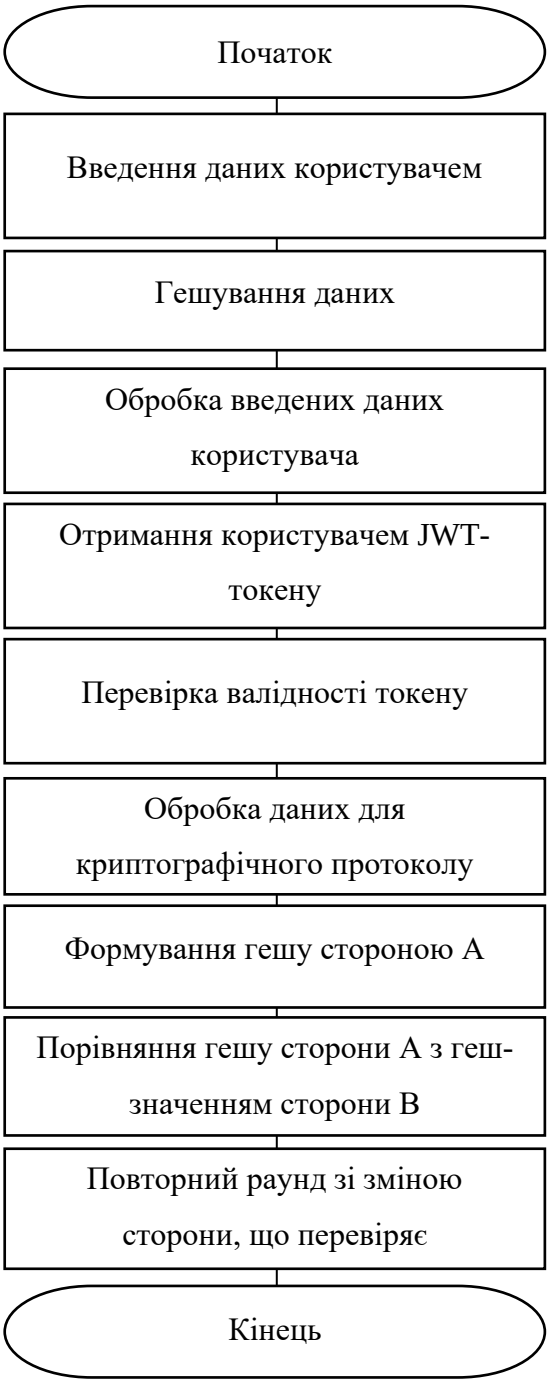
СТРУКТУРА МОДУЛЮ АВТЕНТИФІКАЦІЇ СТОРІН



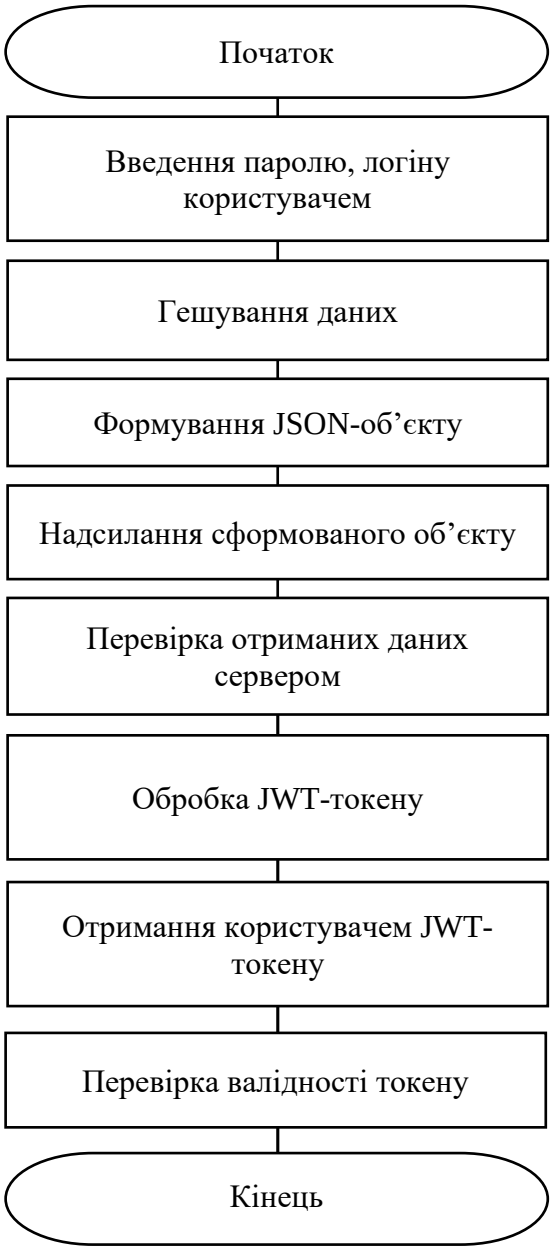
УЗАГАЛЬНЕНИЙ АЛГОРИТМ РОБОТИ ЗАСОБУ



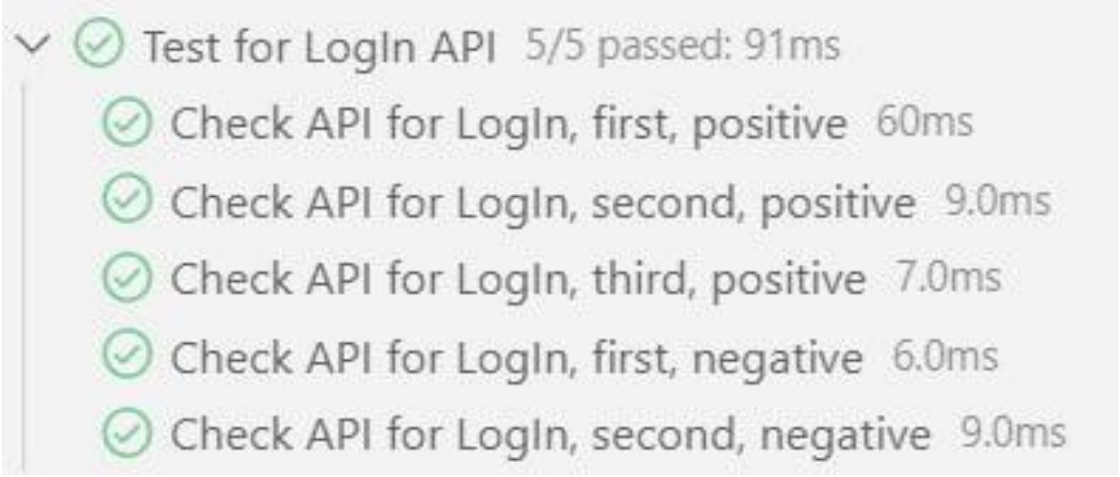
АЛГОРИТМ РОБОТИ БЛОКУ АВТЕНТИФІКАЦІЇ СТОРІН



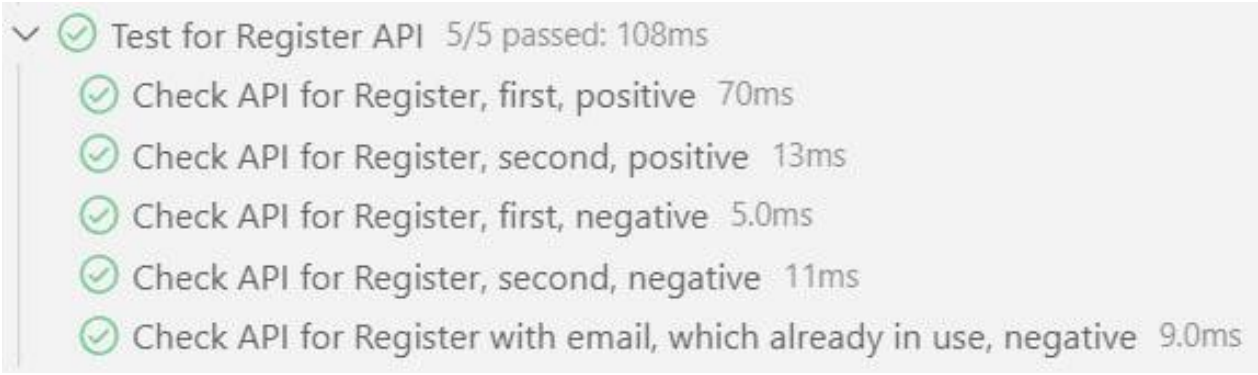
АЛГОРИТМ РОБОТИ БЛОКУ АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ



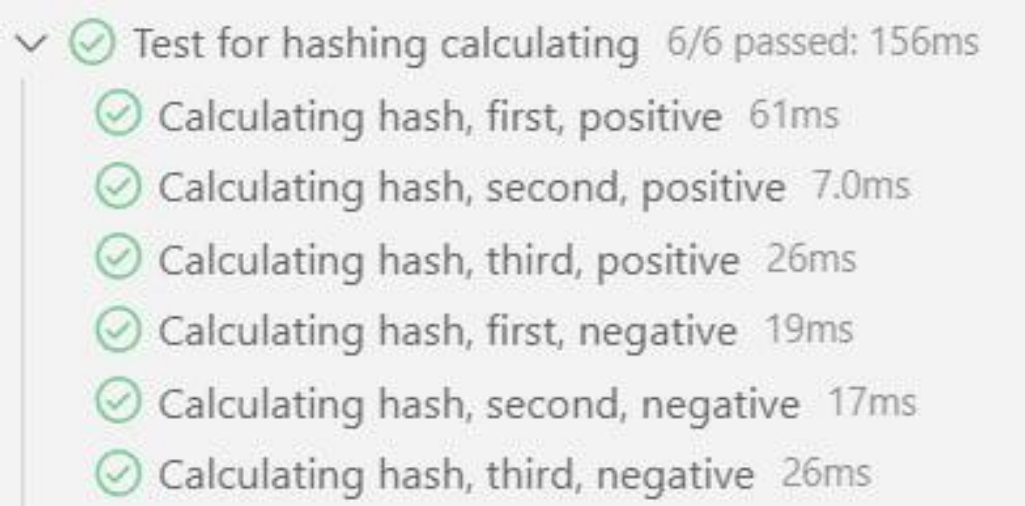
РЕЗУЛЬТАТИ БЛОКОВОГО ТЕСТУВАННЯ



Результати тестування групи Test for Login API

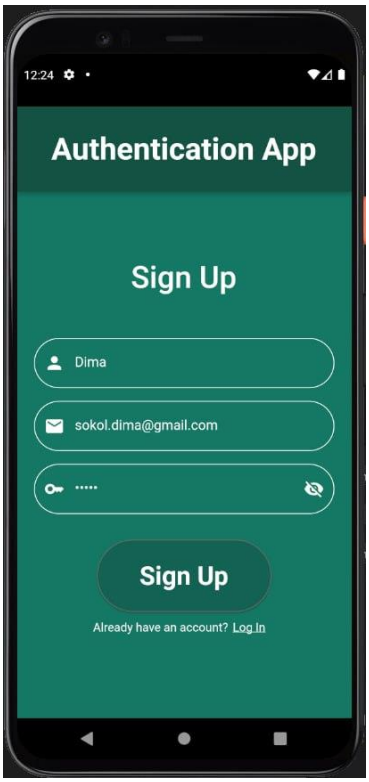


Результати тестування групи Test for Register API

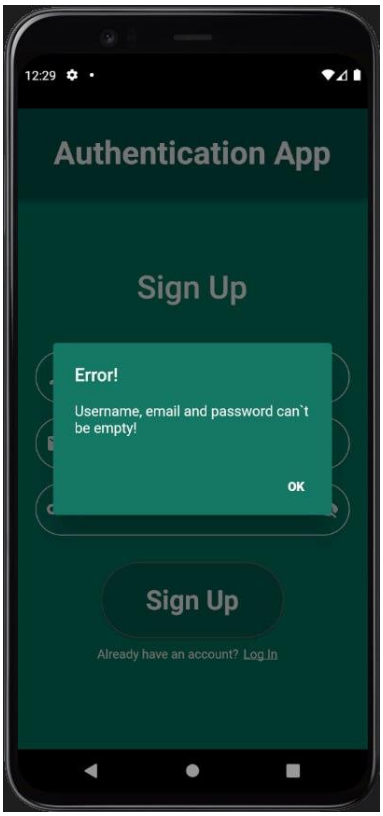


Результати тестування групи Test for hashing calculating

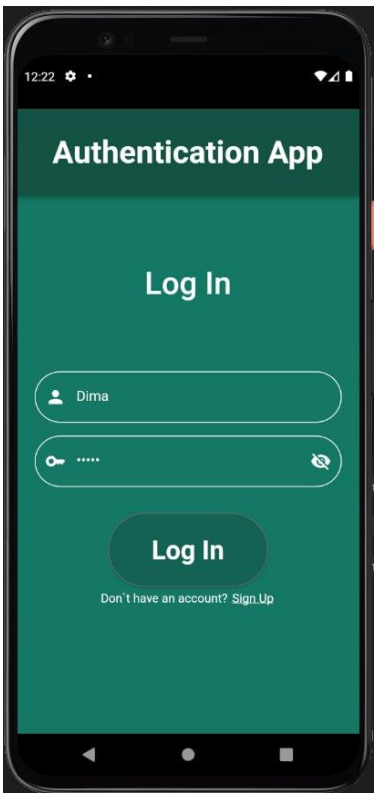
РЕЗУЛЬТАТИ ІНТЕГРАЦІЙНОГО ТЕСТУВАННЯ



Екран проходження реєстрації користувача



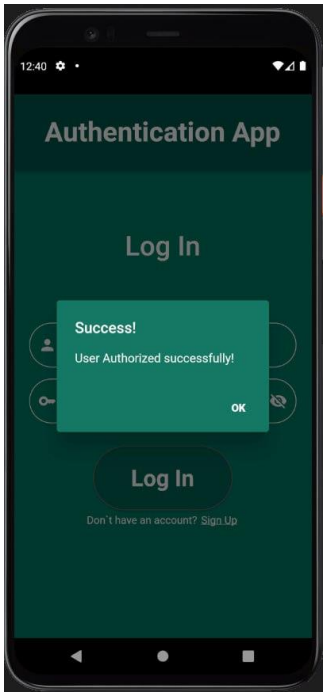
Екран проходження з помилкою про пусті поля



Сторінка автентифікації

Помилка автентифікації через некоректні дані

Помилка автентифікації через пусті поля



Повідомлення про успішну автентифікацію