МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра інформаційних систем

(повна назва кафедри)

ДИПЛОМНА РОБОТА

Розробка месенджера з покращеним алгоритмом наскрізного шифрування для безпечного спілкування користувачів

Виконав студент групи ПМІ-44

спеціальності 122 – комп'ютерні науки

(шифр і назва спеціальності)

Малік Н.Ю

(підпис) (прізвище та ініціали)

Керівник Бернакевич І.Є.

(підпис) (прізвище та ініціали)

Рецензент

(підпис) (прізвище та ініціали)

2023

ЗМІСТ

РЕФЕРАТ

Дипломна робота з теми «Розробка месенджера з покращеним алгоритмом наскрізного шифрування для безпечного спілкування користувачів». Загальний обсяг дипломної роботи складає 48 сторінок.

Предметом дослідження є Месенджер, система обміну повідомленнями через Інтернет, а також технологія, яка використовується для передачі та зберігання конфіденційних даних користувачів**.**

Мета роботи — Розробка месенджеру – веб додатку, який забезпечує захист даних користувача та може успішно протистояти спробам несанкціонованого доступу до зв’язку абонента під час атак типу «людина посередині» за допомогою шкідливого програмного забезпечення, що встановлене на пристрої, на якому запущений клієнт месенджера. Атаки типу "людина посередині" та фізичне захоплення пристроїв користувачів. Використання такого месенджера знижує комфортність спілкування, але підвищує рівень інформаційної безпеки учасників обміну інформацією. Месенджер рекомендується для тих, хто хоче зберегти конфіденційність свого професійного і приватного життя.

ВСТУП

У зв'язку зі зростанням кількості кібератак у всьому світі, питання конфіденційності публічних даних набуває першочергового значення; ті, кому загрожує несанкціонований доступ до їхніх даних через спілкування з іншими коментаторами в Інтернеті, можуть зіткнутися з дуже серйозними наслідками такого розкриття даних, не тільки з точки зору економіки, але й з точки зору загрози життю і здоров'ю людей.

Основною завданням цієї системи є вирішення нелегітивного доступу до даних користувачів, а також підвищення цілісності та конфіденційності цієї інформації. Для досягнення цих цілей в системі використовується шифрування AES-128.

1. Стек використаних технологій для написання месенджера

1.1 JS та React

JavaScript (JS) є мовою програмування, яка широко використовується для створення веб-додатків та веб-сайтів. Вона є однією з найпопулярніших мов у веб-розробці. JS відома своїм високим рівнем взаємодії з користувачем, оскільки може працювати безпосередньо у веб-браузерах, надаючи можливість створювати динамічні та інтерактивні веб-сторінки.

React, з свого боку, є бібліотекою JavaScript для розробки користувацького інтерфейсу. Вона дозволяє розробникам створювати компоненти та перевикористовувати їх, забезпечуючи простоту підтримки та розширення. Основна концепція React - віртуальний DOM (Document Object Model), що дозволяє ефективно оновлювати лише змінені елементи інтерфейсу, замість повного перерендерингу всієї сторінки.

React також використовує компонентний підхід до розробки, де інтерфейс розбивається на незалежні компоненти, які можуть бути повторно використані та управляють своїм станом. Це спрощує створення складних інтерфейсів та поліпшує організацію коду.

Одним з важливих аспектів React є використання JSX (JavaScript XML) - розширення синтаксису JavaScript, що дозволяє описувати інтерфейс у вигляді компонентів, схожих на HTML-код. Це робить код React більш читабельним та зручним для розробників.

Крім того, React має потужну систему керування станом, що дозволяє ефективно управляти даними та їх оновленнями в інтерфейсі. Це робить React ідеальним інструментом для розробки складних додатків зі змінним станом.

Загалом, JavaScript та React є основними технологіями в сучасній веб-розробці, що надають розробникам потужні інструменти для створення динамічних, інтерактивних та ефективних веб-додатків та веб-сайтів.

1.2 NodeJs

Node.js є середовищем виконання JavaScript на серверній стороні, що дозволяє розробникам створювати швидкі та масштабовані серверні додатки. Воно базується на рушії V8, розробленому компанією Google, і забезпечує зручний спосіб розробки, використовуючи одну мову програмування - JavaScript - як для клієнтського, так і для серверного коду. Це сприяє покращенню продуктивності розробки та забезпечує єдність коду між фронтендом та бекендом.

Одним з ключових переваг Node.js є його асинхронна (неблокуюча) архітектура, що дозволяє обробляти багато запитів одночасно без блокування потоку виконання. Це забезпечує високу масштабованість та продуктивність додатків, особливо в ситуаціях з великим навантаженням або в реальному часі.

Крім того, Node.js має широкий вибір зовнішніх модулів, доступних через пакетний менеджер npm. Це дозволяє розробникам використовувати різноманітні засоби та бібліотеки для розробки різної функціональності, такої як мережеві операції, робота з базами даних, розробка API та багато іншого.

Node.js також використовується для створення сучасних серверних фреймворків, таких як Express.js, Nest.js, Koa.js та інші. Ці фреймворки спрощують розробку серверних додатків, надаючи готові рішення для маршрутизації, обробки запитів та інтеграції з різними інструментами.

Використання Node.js дозволяє розробникам створювати ефективні серверні додатки, що здатні обробляти великі обсяги даних та надавати високу продуктивність. Це популярне рішення в сфері веб-розробки та знаходить застосування у різних галузях, включаючи розробку веб-сайтів, API, мікросервісів та багато іншого.

1.3 База даних MongoDB

MongoDB є документно-орієнтованою базою даних, яка використовує схему з варіативною структурою даних. У MongoDB дані зберігаються у вигляді документів, які представляють собою JSON-подібні об'єкти. Це відрізняє її від традиційних реляційних баз даних, де дані організовані у вигляді таблиць і рядків.

Однією з ключових переваг MongoDB є гнучкість та швидкодія роботи зі змінними схемами даних. Документи можуть мати різні структури, що дозволяє зберігати дані без потреби у попередньому визначенні структури таблиць. Це особливо корисно в ситуаціях, коли вимоги до структури даних змінюються часто або коли необхідно зберігати дані різної структури, наприклад, у веб-додатках з динамічними формами.

MongoDB також надає гнучкі можливості для масштабування. Вона підтримує горизонтальне масштабування, що дозволяє розподіляти навантаження на кілька серверів і реплікувати дані для забезпечення високої доступності. Це робить MongoDB популярним в веб-додатках з великим обсягом даних та вимогами до швидкодії.

MongoDB також має потужний запитовий мову - MongoDB Query Language (MQL), який дозволяє виконувати різноманітні операції з даними, включаючи пошук, сортування, фільтрацію та агрегацію. Вона також підтримує індексування даних для поліпшення швидкодії запитів.

MongoDB може використовуватися як самостійна база даних або в поєднанні з іншими технологіями. Вона має багато драйверів для різних мов програмування, що дозволяє легко інтегрувати її зі сучасними додатками.

Загалом, MongoDB є потужною і гнучкою базою даних, яка надає розробникам можливість працювати з документно-орієнтованими даними без обмежень схеми. Вона підтримує масштабування та має багато корисних функцій для роботи з даними. Це робить її популярним вибором для розробки сучасних веб-додатків та систем з великим обсягом даних.

2. Аналіз наявних систем обміну повідомленнями

2.1 Відомості

Сьогодні програми обміну миттєвими повідомленнями є одним з найпопулярніших інструментів для швидкого обміну інформацією. Програми обміну повідомленнями дозволяють миттєво передавати текстову, фото-, відео- та аудіоінформацію, а також відстежувати прочитане.

У цьому розділі розглянуто три найпоширеніші месенджери, проаналізовано їхні функції та рівень інформаційної безпеки.

Програми для обміну повідомленнями - це програмне забезпечення, що використовується для обміну текстовою, голосовою та графічною інформацією через Інтернет між двома або більше абонентами.

Сервер - апаратне забезпечення, призначене для зберігання, отримання та обробки даних, наданих клієнтом або іншим сервером, а також для передачі даних клієнту або іншому серверу.

Клієнт - програмне забезпечення, що працює на пристрої абонента, призначене для обробки та зберігання даних, отриманих від сервера, а також для відправки даних на сервер.

Аутентифікація - процес перевірки автентичності суб'єкта, який намагається отримати доступ до об'єкта з обмеженим доступом.

Авторизація - це процес визначення повноважень суб'єкта впливати на об'єкт.

Ідентифікація - це процес, за допомогою якого суб'єкт або група суб'єктів розпізнається серед інших суб'єктів на основі унікальних характеристик суб'єкта, що розпізнається.

Шифрування - це процес, який здійснює оборотне перетворення інформації, змінюючи її зміст, кількість та значення з метою захисту від несанкціонованого доступу зловмисників під час передачі даних між абонентами. Під час шифрування інформація стає незрозумілою та незчитуваною для тих, хто не має правильного ключа або методу розшифрування. Це забезпечує конфіденційність та безпеку під час комунікації,

дозволяючи тільки авторизованим особам отримувати доступ до зрозумілої інформації.

Розшифрування - процес перетворення інформації із зашифрованої форми у вихідну форму шляхом відновлення її змісту, кількості та значення.

Криптографічний ключ - відкриті або секретні дані, необхідні для шифрування та дешифрування інформації.

Шифротекст - інформація в зашифрованому вигляді без можливості дізнатися початковий зміст для запобігання розшифровування.

Хеш - абсолютне спотворення, при якому значення, кількість і зміст електронної інформації незворотно перетворюється в блок даних фіксованої довжини, значення якого є унікальним і повністю змінюється, якщо інформація змінюється до перетворення.

Конфіденційність - це властивість даних, яка передбачає, що вони залишаються в таємниці і недоступні для несанкціонованих осіб. Це означає, що інформація може бути доступна тільки тим, хто має право на її отримання. Конфіденційність забезпечує, що дані залишаються приватними, недоступними для сторонніх осіб, і можуть бути розкриті лише авторизованим користувачам або суб'єктам, які мають відповідні права доступу.

Цілісність - це властивість даних, що означає відсутність якісних змін, які змінюють зміст, значення або кількість інформації, а також її придатність до використання.

Доступність - це властивість даних, яка означає можливість читати, змінювати, видаляти, зберігати або виконувати деякі з перерахованих вище операцій з інформацією.

2.2 Огляд месенджера Telegram

2.2.1 Можливості, що надає месенджер

За офіційними даними, Telegram наразі використовують понад 700 мільйонів користувачів по всьому світу.

Також він може використовувати GPS для передачі місцезнаходження клієнта. Месенджер також дозволяє користувачам здійснювати відео- та аудіодзвінки між собою. Кожне повідомлення має індикатор статусу, який вказує час передачі повідомлення від відправника до отримувача, а також стадію передачі повідомлення. Повідомлення в приватних чатах можуть бути видалені користувачем як локально, так і односторонньо, без залишення слідів на пристрої отримувача.

Telegram дозволяє користувачам спілкуватися в колі кількох підписників і створювати Telegram-канали. Кожне повідомлення може бути інтерпретоване іншими відвідувачами ресурсу з дозволу адміністратора каналу.

У месенджері також можна створювати секретні розмови, в яких повідомлення автоматично видаляються всіма клієнтами-учасниками через час, визначений творцем.

Telegram - один з небагатьох месенджерів, який дозволяє користувачам створювати і публікувати на каналі різного роду опитування, а також редагувати надіслані повідомлення протягом доби з моменту їх отримання цільовим клієнтом.

2.2.2 Архітектура

Telegram використовує хмарне сховище і клієнт абонента для зберігання даних, що свідчить про те, що архітектура месенджера є клієнт-серверною. Повідомлення надсилаються від абонента А до абонента Б за наступним алгоритмом:

1) Аліса формулює повідомлення та виконує дію, натискаючи кнопку для його відправлення.

2) Повідомлення шифрується за допомогою довготривалого шифру і записується в пам'ять пристрою клієнтом Аліси.

3) Це ж повідомлення шифрується симетричним шифром і відправляється клієнтом Аліси на сервер в Інтернеті.

4) Сервер отримує зашифрований текст і посилає сигнал прийому клієнту Аліси.

5) Сервер запитує клієнта Боба, чи готовий він прийняти повідомлення Аліси. Якщо клієнт підключений до Інтернету, то зашифрований текст надсилається

клієнту Боба через Інтернет. Якщо ні, то коли клієнт Боба знову підключається до інтернету, він запитує сервер, чи було йому надіслано повідомлення. Сервер зберігає дайджест повідомлень для кожного повідомлення.

6) Після того, як зашифрований текст отримано клієнтом Боба, повідомлення розшифровується і Боб може його прочитати.

7) Розшифроване повідомлення клієнт Боба зашифровує довготривалим паролем і зберігає в пам'яті пристрою.

8) Клієнт Аліси підтверджує, що клієнт Боба отримав повідомлення. Після того, як Боб ознайомився з повідомленням, Аліси, він отримує підтвердження про те, що це повідомлення було надіслано.

2.2.3 Захист даних

Захист забезпечується протоколом MTProto, розробленим Telegram.

Передача повідомлень шифрується відкритим ключем з використанням протоколу DiffieHellman, розрахованого за схемою шифрування RSA. Протокол MTProto включає в себе протокол TLS версії 1.3, який забезпечує додаткову безпеку під час передачі повідомлень і обміну повідомленнями.

Слід відмітити, що механізм передачі даних, який використовується в Telegram, має захист від атак "зловмисник посередині" завдяки перевірці автентичності сертифікатів, що обмінюються між клієнтом і сервером під час встановлення з'єднання. Однак, якщо зловмиснику вдасться підробити сертифікат сервера, який не буде виявлений механізмом перевірки, це може призвести до компрометації всієї криптосистеми та порушення конфіденційності передачі даних.

Telegram зберігає реєстраційні дані користувачів у відповідній базі даних на сервері, що дозволяє аутентифікувати та ідентифікувати користувачів з будь-якого пристрою, що підтримує використання додатка. Реєстрація виконується на основі номера SIM-карти, який ідентифікує користувача, а також одноразового пароля (OTP), який надсилається у вигляді SMS-повідомлення на SIM-карту абонента. Це також

слугує формою аутентифікації. Додаток також підтримує можливість автентифікації за допомогою довгострокового пароля, який користувач створює під час входу до свого облікового запису.

2.3 Огляд месенджера Viber

2.3.1 Можливості, що надає месенджер

Viber Media створила однойменний месенджер, який, за офіційними даними компанії, має понад мільярд користувачів.

Функціонал месенджера Viber багато в чому схожий на функціонал додатку Telegram. Однак є і суттєві відмінності:

За словами Viber Media S.à r.l., після відправлення повідомлення, його копія на сервері видаляється і доступ до неї можливий лише у клієнта абонента. У випадку Telegram, всі комунікації зберігаються на сервері протягом певного періоду часу і можуть бути перенесені в файлову систему пристрою абонента, а також надається можливість завантаження цих даних з сервера, якщо відповідні файли не були видалені з файлової системи сервера. Відмінності між Viber і Telegram також полягають у можливості передачі геолокації. Viber дозволяє передати геолокацію клієнта в один момент часу, тоді як в Telegram відсутня можливість транслювати геолокацію протягом фіксованого періоду часу. Обидва месенджера дозволяють надсилати конфіденційні повідомлення, які автоматично видаляються через певний проміжок часу. Але відмінність полягає в тому, що можливість видалення повідомлень на клієнті автора і співрозмовника відрізняється. Існує факт видалення повідомлень обома сторонами, який може бути усунутий лише в разі повного видалення розмови.

2.3.2 Архітектура

Архітектура Viber месенджера дуже схожа на відповідні параметри Telegram. Однак, на відміну від Telegram, розробники Viber стверджують, що жодним чином не зберігають на своїх серверах комунікації користувачів після того, як повідомлення дійшло до адресата. Тому, якщо месенджер одного і того ж облікового запису видалити на всіх пристроях абонента, а потім перевстановити додаток, відновиться

лише інформація про те, що абонент брав участь у чаті. Зміст розмови або чату з абонентом повністю видаляється. З цієї ж причини, якщо один або декілька файлів, надісланих абоненту, будуть видалені локально з файлової системи терміналу, відновити вміст файлів шляхом їх повторного завантаження буде неможливо, навіть якщо у партнера по чату на терміналі є вищевказані дані. Однак факт передачі файлу збережеться.

Viber після видалення та перевстановлення додатку може синхронізувати його з іншими клієнтами цього ж облікового запису на відповідному пристрої того ж Абонента або, за бажанням користувача, створити комунікацію з резервною копією даних, яка згодом зберігатиметься на серверах Google, забезпечивши таким чином можливість збереження даних. У зв'язку з цими обставинами компанія наголошує, що не гарантують конфіденційність даних користувача під час створення резервних копій повідомлень, і не несуть за це відповідальності.

2.3.3 Захист даних

Viber вживає комплексний набір заходів для захисту даних користувачів. Ось детальна інформація про захист даних у Viber:

Кінцеве до кінця (End-to-End) шифрування повідомлень: Viber використовує протоколи шифрування, які забезпечують кінцеве до кінця шифрування повідомлень. Це означає, що повідомлення шифруються на пристрої відправника і розшифровуються тільки на пристрої отримувача. Навіть сама компанія Viber не має можливості переглядати розшифрований зміст повідомлень.

- Шифрування даних у спокої (at rest): Дані, що зберігаються на серверах Viber, також шифруються. Це означає, що навіть якщо зловмисники отримають доступ до серверів, дані будуть захищені шифруванням і недоступні для них.

- Двоетапна перевірка: Viber надає можливість встановити двоетапну перевірку для облікового запису. Це додатковий шар захисту, який вимагає введення додаткового підтвердження, наприклад, через SMS-повідомлення або аутентифікатор додаток, для доступу до облікового запису.
- Захист від злому паролів: Viber використовує механізми захисту, які унеможливлюють злом паролів користувачів та не дозволяють компанії отримувати доступ до паролів.
- Захищений доступ до даних: Viber вживає заходів для захисту доступу до даних користувачів, використовуючи шифрування та механізми контролю доступу.
- Аудит безпеки: Команда Viber проводить регулярний аудит безпеки своєї системи з метою виявлення та виправлення потенційних вразливостей.
- Захист приватності: Viber дотримується політики приватності та вимог щодо збору та використання персональних даних користувачів. Компанія не рекламує особисті дані користувачів третім особам без згоди користувача.

Важливо зазначити, що хоча Viber робить значні зусилля для захисту даних користувачів, безпека залежить від кінцевих користувачів. Користувачам слід приділяти увагу заходам безпеки, таким як використання міцних паролів, оновлення програмного забезпечення та уважне ставлення до розкриття особистої інформації.

Видалення Messenger і перевстановлення програми назавжди видаляє вміст повідомлень, що підвищує рівень конфіденційності. Секретні повідомлення можуть надсилатися всім клієнтам всіх комунікаційних акаунтів, які видаляються через певний проміжок часу, що знижує ризик втрати переданої інформації або порушення конфіденційності.

2.4 Висновки щодо месенджерів

Результати аналізу показують, що використання месенджерів може призвести до порушення конфіденційності та цілісності даних користувачів, особливо якщо зловмисник отримає доступ до пристрою з встановленим додатком. Несанкціонований доступ до сервера також може призвести до розкриття як минулих, так і майбутніх повідомлень користувачів.

Viber, наприклад, застосовує систему шифрування для захисту даних користувачів і ефективно бореться зі спробами порушити конфіденційність і цілісність інформації. Однак, він має певні недоліки. Наприклад, при атаках типу "людина посередині" зловмисники мають можливість використовувати підроблені сертифікати сервера, які неможливо відрізнити від справжніх сертифікатів для клієнта. Це може створити потенційну загрозу для конфіденційності та цілісності даних, а також погрозу компрометації криптографічних ключів.

У разі Viber, месенджер може протистояти крадіжці даних при крадіжці пристрою користувача з SIM-картою. Проте, OTP-паролі можуть бути зчитані заблокованого пристрою при вимкненому екрані. Крім того, використання сканера QR-коду для верифікації автентичності може бути компрометоване.

Застосування TLS версії 1.2 знижує рівень захисту в цілях забезпечення сумісності месенджера з різними операційними системами та їх версіями. Резервні копії також зберігаються незахищеними на серверах Google, що підвищує ризик втрати інформації. Зловмисники також можуть отримати повний доступ до комунікацій користувача, клонуючи SIM-карту, видаляючи додатки на пристрої абонента або вимикаючи пристрій.

3. Аналіз методів, засобів та протоколів захисту в системах обміну повідомленнями.

3.1 Протоколи передачи даних та їх аналіз

3.1.1 Протоколи TCP/IP

Залежно від використовуваних потреб, існують різні протоколи передачі даних, які мають різні характеристики. Один з цих протоколів - TCP/IP, є ключовим для передачі даних і відіграє важливу роль у застосуванні криптографічних протоколів передачі.

Процес передачі даних в стеку TCP/IP базується на наступних етапах:

1. Клієнт починає процес з'єднання, передаючи серверу спеціальне службове повідомлення, яке містить стартовий номер послідовності пакетів (ISN), синхронізуючий біт (SYN) і вказує порти клієнта і сервера для цього повідомлення.
2. Сервер отримує повідомлення від клієнта, запам'ятовує номер послідовності та відповідає повідомленням з встановленими бітами SYN і ACK. Відповідь сервера також містить номер ISN, який збільшений на одиницю. Крім того, сервер передає свій номер послідовності байтів, які ідентифікують окремі байти в повідомленнях, щоб уникнути переплутання даних між різними з'єднаннями.
3. Клієнт підтверджує отримання відповіді сервера, відправляючи повідомлення ACK з номером послідовності, який згенерував сервер і збільшений на одиницю. Клієнт також вказує номер байту даних, який очікується надіслати до сервера.
4. Передача даних відбувається шляхом включення сегментів даних, номерів їх місця в файлах та номерів послідовності пакетів під час передачі.
5. Завершення з'єднання ініціюється сервером, який надсилає повідомлення клієнту з бітами завершення (FIN) і ACK.
6. Клієнт також розриває з'єднання, відправляючи повідомлення з бітами ACK і FIN.

3.1.2 Безпечний протокол Transport Layer Security

TLS (Transport Layer Security) є протоколом передачі даних, який забезпечує захищене шифрування повідомлень та міжособистостне шифрування даних, що дозволяє запобігти несанкціонованому доступу та перехопленню інформації. Існує три версії цього протоколу: TLS 1.1, TLS 1.2 і TLS 1.3 [6]; TLS 1.3 є попередньою версією TLS 1.2 і TLS 1.1 з додатковими можливостями і оптимізованими алгоритмами, тому нижче описано механізм останньої версії.

Процедура встановлення з'єднання має схожий принцип як у стеку TCP/IP, проте під час обміну відбувається передача іншої інформації та службових повідомлень, які ініціюють саме з'єднання.

1) Клієнт, який намагається з'єднатися з сервером, надсилає повідомлення, що містить наступну інформацію.

- Версія протоколу TLS,
- 32-байтне випадкове значення для клієнта, щоб запобігти підробці часу на вузлах мережі,
- Ідентифікатор сеансу: для перевірки раніше встановленого сеансу між сервером і клієнтом,
- Список паролів, що підтримуються клієнтом. Порядок пунктів визначає порядок переваг, в якому клієнт хоче використовувати той чи інший шифр (перший є найбільш бажаним),
- Список методів стиснення даних, що підтримуються клієнтом,
- Дані про деякі розширення протоколу TLS, що дозволяють використовувати нові технології шифрування і передачі даних, які стали доступні після затвердження протоколу TLS.
- Кожне поле перед вмістом має власне значення довжини.

2) Після отримання запиту, сервер реагує, надсилаючи повідомлення про помилку, яке містить деталі про виниклу помилку, або відправляє відповідне службове повідомлення з метою встановлення з'єднання:

- Версія протоколу TLS,
- Випадкове 32-байтне значення сервера,
- ідентифікатор сесії,
- набір паролів, обраних сервером зі списку клієнтів, - метод стиснення даних, обраний сервером,
- дані деяких розширень протоколу TLS.

Встановлюється алгоритм для обчислення коду автентифікації повідомлення (MAC), а також обмінюється 48-байтовий головний секретний ключ, який відомий обом авторизованим учасникам з'єднання. Відбувається узгодження відкритого ключа сервера і головного секретного ключа, відомого обом авторизованим учасникам обміну. Далі надсилається сертифікат сервера, що містить відкритий ключ сервера та інші сертифікати, які підтверджують справжність початкового сертифіката.

Далі надсилається повідомлення, що містить дані сервера, які є частиною інформації, необхідної для генерації спільного сеансового ключа. У даному повідомленні передаються параметри протоколу Diffie-Hellman, тимчасовий RSA-ключ та підпис сервера, який використовує систему RSA або ECDSA з використанням хеш-функцій MD5 і SHA-1. Після цього сервер відправляє запит на отримання клієнтського сертифіката, який містить перелік підтримуваних сервером типів і шифрів сертифікатів, а також алгоритм підпису і хеш-функцію.

Це останнє повідомлення в черзі повідомлень-відповідей клієнту є даними, щодо яких сервер має підстави вважати, що всі ініціалізаційні дані надіслано.

3) Після отримання повідомлення від сервера, клієнт перевіряє його автентичність шляхом перевірки підпису з використанням отриманого від сервера відкритого ключа. Клієнт також передає серверу необхідний для автентифікації сертифікат, який містить 48-байтовий випадковий приватний ключ RSA, зашифрований відкритим ключем

клієнта, або відкритий ключ, що згенерований клієнтом за допомогою алгоритму Діффі-Хеллмана. Крім того, клієнт підписує цей послідовний ланцюжок повідомлень.

Потім на сервер надсилається повідомлення з підтвердженням довіри.

У наступному повідомленні клієнт інформує сервер про вибір шифру і про те, що всі наступні повідомлення будуть зашифровані відповідно до домовленості.

Останнє ініціалізаційне повідомлення від клієнта містить підсумок усіх попередніх ініціалізаційних повідомлень поточного сеансу, щоб сервер міг перевірити автентичність ініціалізації. Це повідомлення надсилається в зашифрованому вигляді.

4) Якщо перевірка пройшла успішно, сервер надсилає клієнту сигнал про те, що подальші обміни захищені відповідною криптографією і що ініціалізація завершена.

5) Відбувається безпечний обмін даними. Кожне повідомлення має зашифрований текст і MAC-код для автентифікації повідомлення.

Недоліки.

- Не всі поширені браузери підтримують цю версію протоколу.

Переваги.

- Вища швидкість передачі даних.
- Підвищений рівень безпеки.
- Гнучкість переходу на старішу версію протоколу, якщо абонент не підтримує TLS 1.3;

Відносно проста конфігурація знижує ймовірність вразливостей безпеки через неправильне налаштування..

3.1.3 Безпечний протокол Secure Socket Layer

Secure Socket Layer є криптографічним протоколом, метою якого є забезпечення захищеного обміну інформацією. Основна функція SSL полягає в автентифікації клієнта та шифруванні передаваних даних. Цей протокол працює незалежно від прикладного рівня, що є його перевагою. Після успішної автентифікації клієнта, SSL встановлює алгоритм шифрування та сесійний ключ для

безпечної передачі даних між клієнтом і сервером.

Залежно від налаштувань протоколу, шифрування може бути симетричним або асиметричним.

Процес роботи протоколу SSL включає наступні кроки:

1) Клієнт надсилає запит на з'єднання серверу, включаючи інформацію про потребу в новому майстер-ключі, підтримувані шифри та запит на сертифікат сервера. Якщо новий майстер-ключ не потрібний, переходиться до наступного кроку. У противному випадку клієнт запитує майстер-ключ сервера.

2) Сервер підтверджує запит на з'єднання, відправляючи ідентифікатор сесії, свій сертифікат та список підтримуваних шифрів. Коли виникає потреба у створенні нового шифру, сервер передає свій публічний ключ і запитує у клієнта сертифікат.

3) Потім клієнт проводить чек сертифікату сервера та повідомляє про результати. Крім того, він може надіслати серверу обраний ним алгоритм шифрування, що базується на спільному секреті, та передати новий зашифрований майстер-ключ з використанням відкритого ключа сервера. Клієнт також надсилає серверу свій власний сертифікат. Після цього сервер перевіряє сертифікат клієнта в процесі обміну повідомленнями.

4) Починається подальший захищений обмін інформацією, який ґрунтується на використанні симетричного ключа. Симетричні ключі створюються з майстер-ключа.

Переваги протоколу SSL включають:

• Сумісність з різноманітними пристроями та більшістю програмного забезпечення для передачі даних через мережу.

• Сумісність з технологією віртуальної приватної мережі (VPN).

Недоліки протоколу SSL включають:

- Обмежена підтримка сучасних веб-технологій.
- Можливість легкої підміни сертифікату.

3.1.4 Протокол WSS

Протокол WSS (WebSocket Secure) є розширенням протоколу WebSocket, що забезпечує безпеку зв'язку між клієнтом і сервером. Використання протоколу WSS дозволяє захистити передавані дані шляхом використання SSL/TLS для шифрування та аутентифікації.

Це дає можливість забезпечити конфіденційність та цілісність даних під час їх передачі. Протокол WSS є особливо важливим для застосунків, що обробляють чутливу інформацію, оскільки він забезпечує безпеку комунікації в режимі реального часу через WebSocket.

Протокол WSS (WebSocket Secure) надає декілька переваг:

1. Конфіденційність даних: Усі дані, які передаються між клієнтом і сервером, шифруються, що дозволяє запобігти прослуховуванню або перехопленню даних під час їх передачі.

2. Цілісність даних: Протокол WSS використовує хеш-функції та підписи для перевірки цілісності передаваних даних. Це дозволяє виявити будь-які зміни або спотворення даних під час їх транзиту.

3. Аутентифікація сервера: Використання SSL/TLS забезпечує аутентифікацію сервера за допомогою сертифікатів. Це дозволяє клієнтам перевірити, що вони спілкуються з довіреною стороною та уникнути атак "Man-in-the-Middle".

4. Реальний час: Протокол WSS зберігає всі переваги протоколу WebSocket, забезпечуючи можливість двостороннього обміну даними між клієнтом і сервером у режимі реального часу.

Використання протоколу WSS особливо важливе для додатків, що обробляють конфіденційну інформацію, наприклад фінансові дані, особисту інформацію користувачів або медичні записи. Він забезпечує безпеку та захищеність під час зв'язку між клієнтом і сервером через протокол WebSocket.

3.1.5 Висновки

Після проведення аналізу протоколу передачі даних у месенджері, була обрана версія TLS 1.3. Таке рішення прийнято з метою вибору TLS 1.3, оскільки ця версія протоколу має найбільшу сумісність з криптографічними системами та забезпечує високий рівень захисту від атак типу "людина посередині".

3.2Дослідження шифрів які застосовуваються в месенджерах для безпеки

3.2.1 Алгоритм RSA

Алгоритм RSA (Rivest-Shamir-Adleman) є одним з найбільш поширених асиметричних криптографічних алгоритмів, який використовується для шифрування, розшифрування та підпису даних. Його названо на честь його творців — Рональда Райвеста, Аді Шаміра та Леонарда Адлемана. Алгоритм RSA заснований на труднощах факторизації великих цілих чисел.

Принцип роботи алгоритму RSA:

1. Крок генерації ключів:

- Виберіть два різних простих числа, назвемо їх `p` і `q`.
- Обчисліть їх добуток `n = p * q`, який буде використовуватися як модуль для шифрування і розшифрування.
- Обчисліть значення функції Ойлера для `n`: `phi(n) = (p - 1) * (q - 1)`, де `phi` - функція Ойлера.
- Необхідно обрати ціле число `e`, яке задовольняє умову 1 < `e` < `phi(n)`, а також є взаємно простим з `phi(n)`. Це число `e` буде використовуватися як публічний ключ для шифрування.
- Обчисліть обернене до `e` модуля `phi(n)`, позначимо його як `d`. `d` буде приватним ключем розшифрування.

2. Шифрування:

- Для шифрування повідомлення `M` використовується публічний ключ `(e, n)`.
- Обчисліть шифроване повідомлення `C` за формулою: `C = M^e mod n`, де `^` позначає піднесення до степеня, а `mod` — операцію модуля.

3. Розшифрування:

- Для розшифрування шифрованого повідомлення `C` використовується приватний ключ `d`.
- Обчисліть розшифроване повідомлення `M` за формулою: `M = C^d mod n`, де `^` позначає піднесення до степеня, а `mod` — операцію модуля.

Рисунок 3.1— Опис принципу функціонування алгоритму RSA.

Переваги алгоритму RSA:

• Безпека: RSA базується на складності факторизації великих чисел, що робить його стійким до атак шляхом перебору.

• Асиметричний: RSA використовує пару ключів — публічний та приватний, що дозволяє використовувати їх для шифрування та розшифрування повідомлень.
• Підписи: RSA може бути використаний для створення цифрових підписів, що дозволяє перевіряти автентичність повідомлень.

Недоліки алгоритму RSA:
• Обчислювальна складність: Розшифрування або підписання повідомлень за допомогою приватного ключа RSA може бути обчислювально витратним при великих значеннях ключа.
• Великі розміри ключів: RSA вимагає використання довгих ключів для забезпечення безпеки, що може призводити до більшої виконавчої складності та обсягу передаваних даних.
• Вразливість до побічних каналів: RSA може бути вразливим до атак, що використовують побічні канали, такі як аналіз часу виконання або виток інформації про використані ключі.

Все життєво важливою є відповідна реалізація і використання алгоритму RSA, з урахуванням найновіших рекомендацій з криптографічної безпеки.

3.2.2 Алгоритм Діфі-Хелмана

Алгоритм Діффі-Хелмана є одним з перших публічних ключових протоколів, що використовується для безпечного обміну секретним ключем через незахищену комунікаційну мережу. Цей алгоритм був розроблений Уітфілдом Діффі та Мартіном Гелманом у 1976 році і він дозволяє двом абонентам, які не мають попередньо обговорених ключів, встановити спільний секретний ключ безпосередньо через відкритий канал зв'язку.

Основна ідея алгоритму Діффі-Хелмана базується на складності обчислення оберненого елемента в арифметичних операціях над скінченним

полем. Цей алгоритм використовує математичну проблему дискретного логарифмування для забезпечення безпеки.

Основні кроки алгоритму Діффі-Хелмана:

- Вибір параметрів: кожен з абонентів вибирає випадкове просте число `p` та генератор `g` скінченної групи.
- Обмін публічними ключами: кожен абонент обчислює свій публічний ключ, який він передає іншому абоненту.
- Обчислення спільного секретного ключа: кожен абонент, отримавши публічний ключ іншого абонента, обчислює спільний секретний ключ.

Переваги алгоритму Діффі-Хеллмана:

- Безпека: основана на складності обчислення дискретного логарифму, що забезпечує високий рівень безпеки.
- Публічний обмін ключами: абоненти можуть обмінюватися публічними ключами через незахищені канали, що робить алгоритм ефективним у реальних умовах.

Недоліки алгоритма Діффі-Хелмана:

- Потреба в автентифікації: сам по собі алгоритм не надає механізму для автентифікації абонентів. Для забезпечення цього необхідно використовувати додаткові механізми, наприклад, цифрові підписи.
- Уразливість до атак типу "людина посередині": атакувач може проникнути в комунікаційний канал і підробити ключі, що призводить до викриття конфіденційності та цілісності даних.
- Використання великих чисел: для забезпечення безпеки алгоритму необхідно використовувати великі прості числа, що може призвести до високої обчислювальної складності та споживання ресурсів.

Незважаючи на деякі недоліки, алгоритм Діффі-Хелмана є одним з важливих криптографічних протоколів, що використовуються для безпечного обміну ключами в багатьох протоколах та системах забезпечення інформації.

3.2.3 Алгоритм AES

AES (Advanced Encryption Standard) є симетричним блочним алгоритмом шифрування, який використовується для захисту конфіденційності даних. Цей алгоритм був прийнятий у 2001 році і став стандартом шифрування для урядових та комерційних застосувань.

Основні характеристики алгоритму AES:

1. Блочне шифрування: Advanced Encryption Standard шифрує дані в блоками фіксованого розміру (128 біт).
2. Для шифрування та дешифрування використовується один секретний ключ.
3. Ключовий розмір: підтримує ключі довжиною 128, 192 та 256 біт.
4. Розмір блоку: шифрування проводиться над блоками даних фіксованого розміру (128 біт).
5. Кількість раундів: залежно від розміру ключа, кількість раундів шифрування може бути різною (10 раундів для 128-бітного ключа, 12 раундів для 192-бітного ключа, 14 раундів для 256-бітного ключа).

Переваги алгоритму AES:

- Безпека: AES є одним з найбільш безпечних алгоритмів шифрування. Він витримує багато відомих атак, включаючи перебор ключа і атаки типу "людина посередині".
- Швидкодія: AES має високу швидкодію шифрування і розшифрування на сучасних обчислювальних пристроях.

- Широке застосування: AES є широко використовуваним стандартом і підтримується багатьма криптографічними бібліотеками та програмним забезпеченням.

Недоліки алгоритму AES:

- Використання симетричного ключа: вимагає обміну секретним ключем між відправником і отримувачем, що може бути проблематичним у деяких сценаріях, особливо якщо потрібно безпечно передати ключ на велику відстань.
- Обмежений розмір блоку: AES працює з фіксованим розміром блоку (128 біт), що може призвести до проблем з обробкою довгих повідомлень.

Незважаючи на деякі недоліки, AES залишається надійним та ефективним алгоритмом шифрування, який широко застосовується для захисту конфіденційності даних у різних сферах, включаючи комунікації в мережах, зберігання даних та криптографічні протоколи.

3.2.4 Висновки

RSA, AES і Діффі-Хелмана - це три різних алгоритми шифрування, які мають свої унікальні особливості і застосування.

RSA (Rivest-Shamir-Adleman) - це асиметричний криптографічний алгоритм, який базується на складності факторизації великих цілих чисел. RSA використовує два ключі: публічний ключ для шифрування даних і приватний ключ для розшифрування. Цей алгоритм використовується для шифрування та підпису даних, забезпечуючи конфіденційність та аутентичність.

AES (Advanced Encryption Standard) є симетричним блочним шифром, який знаходить широке застосування в різних сферах. Він використовує один і той же ключ як для шифрування, так і для розшифрування даних. Цей криптографічний алгоритм забезпечує захист конфіденційної інформації, такої як паролі та важливі дані.

Алгоритм Диффі-Хелмана - це протокол обміну ключами, який дозволяє двом або більше сторонам безпечно обмінюватися секретним ключем через незахищену комунікаційну мережу. Він базується на складності обчислення дискретного логарифма і забезпечує конфіденційність даних, які обмінюються між сторонами.

Усі ці алгоритми мають свої переваги і недоліки і використовуються у різних ситуаціях залежно від вимог до безпеки і швидкодії. RSA часто використовується для шифрування малих об'ємів даних, підпису та обміну ключами. AES застосовується для шифрування великих об'ємів даних, таких як файли і повідомлення. Алгоритм Диффі-Хелмана використовується для безпечного обміну ключами в розподілених системах.

Висновок полягає в тому, що вибір алгоритму шифрування залежить від конкретних потреб і контексту, а також від рівня безпеки, швидкодії і ресурсів, доступних для реалізації алгоритму. RSA, AES і Диффі-Хелмана є важливими криптографічними інструментами, які використовуються для захисту конфіденційності, цілісності та аутентичності даних.

3.3 Дослідження хеш функцій

3.3.1 Криптографічна хеш-функція SHA-256

SHA-256, також відомий як Secure Hash Algorithm 256-bit, є криптографічною хеш-функцією, що використовується для обчислення унікального хеш-значення повідомлення. Робота SHA-256 базується на блочній криптографії, де вхідне повідомлення розбивається на блоки фіксованого розміру та обробляється послідовно кожен блок.

Процес SHA-256 складається з наступних кроків:

1. Ініціалізація початкових значень: Встановлюються початкові значення, відомі як початковий вектор стану (IV) та початкові константи.
2. Підготовка повідомлення: Вхідне повідомлення доповнюється до необхідної довжини та розбивається на блоки однакового розміру.
3. Ініціалізація стану: Стан SHA-256 складається з вектора регістрів, які початково ініціалізуються початковими значеннями.
4. Обчислення хеш-значення: Кожен блок повідомлення обробляється послідовно за допомогою ітераційного процесу, який включає блочні операції, такі як перестановки бітів, побітові логічні операції та арифметичні операції.
5. Формування хеш-значення: Після обробки всіх блоків повідомлення, фінальний стан SHA-256 перетворюється в унікальне хеш-значення. Це хеш-значення служить ідентифікатором для конкретного вхідного повідомлення.

Основна характеристика SHA-256 полягає в тому, що навіть незначні зміни в початковому повідомленні призводять до суттєвих змін у вихідному хеш-значенні. Це робить його використовуваним для перевірки цілісності даних та створення цифрових підписів.

SHA-256 є одним з найпоширеніших алгоритмів хешування, що застосовуються у різних галузях, включаючи криптографію, мережеву безпеку, блокчейн та інші. Він забезпечує надійний та ефективний спосіб генерації унікальних хеш-значень для повідомлень будь-якого розміру.

3.3.2 Хеш-функція bcrypt

bcrypt.hash є функцією, що використовує алгоритм bcrypt для хешування паролю. Хешування - це процес перетворення вхідних даних, в даному випадку паролю, в

короткий, незворотний хеш. Хешовані значення неможливо перетворити назад у вихідні дані, що робить їх корисними для зберігання і порівняння паролів без зберігання самого паролю.

У випадку bcrypt.hash, вхідними параметрами є пароль і значення "солі" (`salt`). "Сіль" (salt) - це випадкова дана, яка додається до вхідних даних перед хешуванням. Вона використовується для ускладнення процесу хешування і підвищення безпеки паролю.

Хешування паролю за допомогою bcrypt включає в себе кілька раундів обчислень, що робить його більш тривіальним для обчислення, ніж простіші алгоритми хешування. Кількість раундів визначається значенням солі. У випадку `bcrypt.hash(password, salt, (err,hash)`, значення `salt` вказує на кількість раундів, які будуть виконані під час хешування.

Результатом виклику `bcrypt.hash` є хешоване значення паролю, яке зберігається в змінній `hash`. Цей хеш може бути використаний для порівняння збереженого хешу з введеним користувачем паролем для перевірки його правильності без збереження самого паролю.

Важливо враховувати, що bcrypt є міцним алгоритмом хешування, який добре відповідає вимогам безпеки для зберігання паролів..

3.3.3 Висновки

Залежно від контексту використання, різні хеш-функції можуть мати свої переваги та недоліки.

SHA-256 є сильною та безпечною хеш-функцією, яка забезпечує високий рівень стійкості до атак перебором. Вона широко використовується у багатьох криптографічних протоколах та застосунках.

MD5, з іншого боку, є застарілою хеш-функцією, яка більш вразлива до атак злому. Вона залишає можливість для колізій, коли два різні вхідні повідомлення можуть мати один і той самий хеш.

bcrypt є алгоритмом хешування, спеціально розробленим для збереження паролів із підтримкою солей та медленного хешування. Він відносно повільний, що робить його більш стійким до атак перебором.

Отже, якщо ми маємо на увазі безпеку паролів, bcrypt є більш перевагою над SHA-256 та MD5. Однак, якщо йдеться про широкий спектр криптографічних застосунків, SHA-256 буде більш популярним та рекомендованим вибором.

4. Упорядкування системи та структури інформації

4.1 Структура передачі даних між учасниками.

Взаємодія між двома абонентами включає участь трьох сторін: сторони alpha, сторони beta та сервера. Сервер виступає посередником у передачі зашифрованого ключа для шифрування між абонентами. Він використовує WebSocket для передачі ключа шифрування який в свою чергу зашифрований алгоритмом AES-256, що дозволяє зберегти конфіденційність ключа. WebSocket передає дані і транспортує їх через захищене з'єднання. Також сервер використовує тимчасове сховище даних під час обміну повідомленнями.

4.2 Методи збереження листування в додатку та процедури автентифікації

Код відображає функціонал листування між абонентами. При надсиланні повідомлення відбувається генерація випадкового ключа шифрування AES-128.

Після чого відбувається шифрування самого ключа за допомогою алгоритма AES-256.

Текст повідомлення шифрується за допомогою отриманого ключа та надсилається на сервер разом з зашифрованим ключем.

Зберігають повідомлення на сервері та шифрують текст повідомлень з використанням ключа шифрування AES-128.

Під час отримання повідомлення спочатку декодується ключ і після розшифровується повідомлення.

4.3 Клієнтська частина та UI

Месенджер дозволяє авторизацію користувачів у системі. Тобто перше, що бачe клієнт це віконце логіну або реєстрації. Після успішної реєстрації, користувач

може залогінитись в меню “Login”, після чого може розпочати обмін повідомленнями з іншим користувачем.

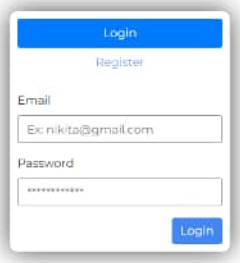


Рисунок 4.1 — Форма логування користувача

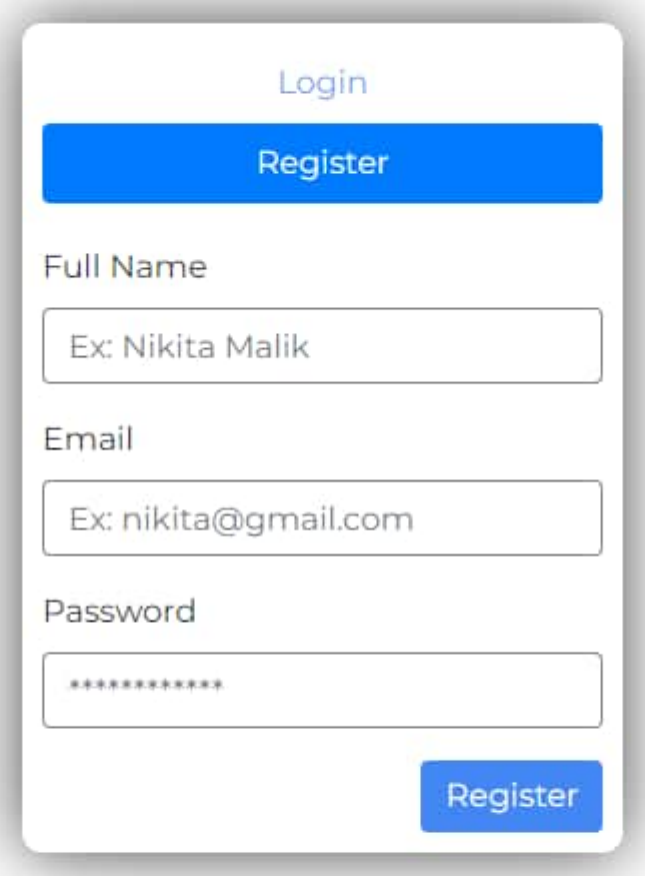
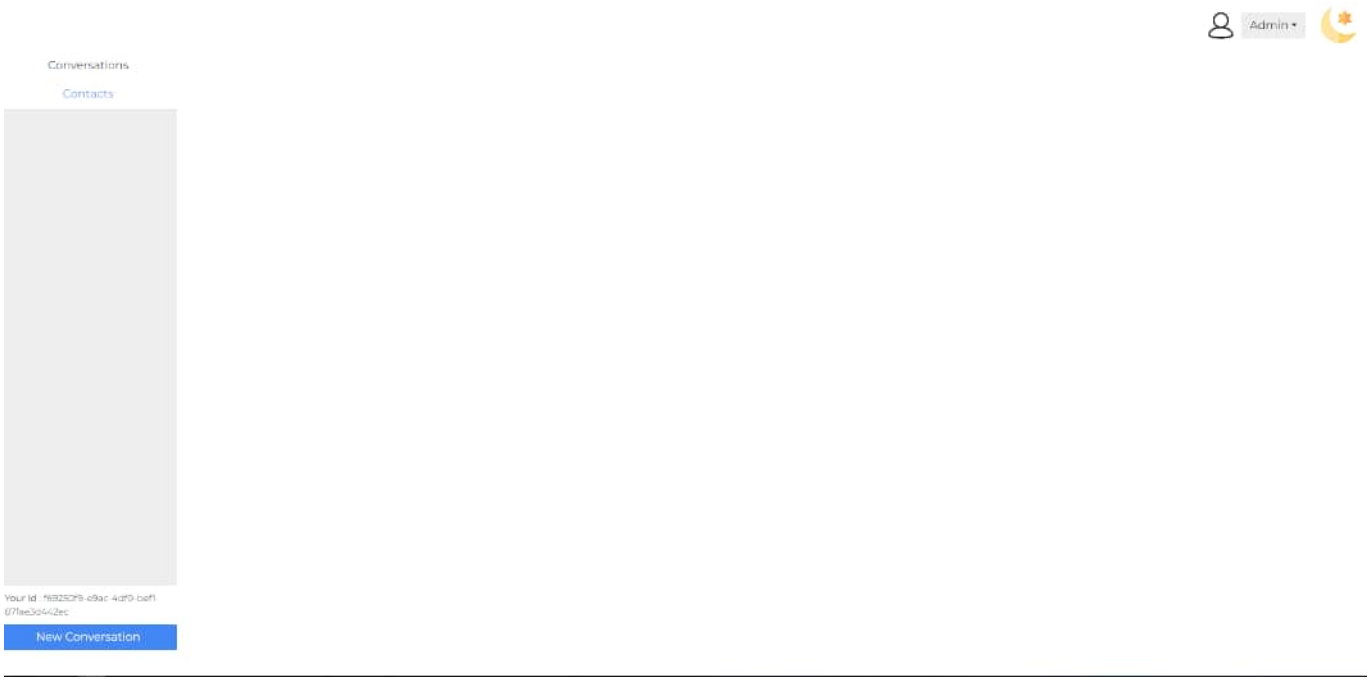


Рисунок 4.2 — Форма реєстрації користувача

Після реєстрації та логіну нас вітає головна сторінка, вона досить мінімалістична, також можна побачити, що зліва знизу в нас є наш Id, цей Id формується при реєстрації користувача і він унікальний для кожного

користувача, за його допомогою ми будемо додавати контакти.



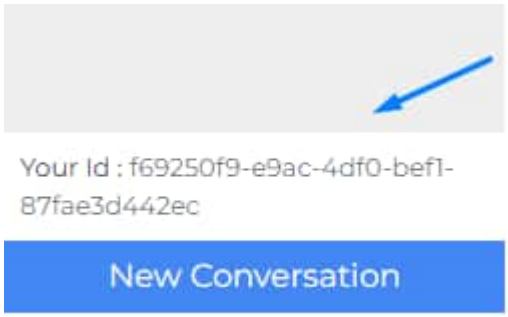


Рисунок 4.3 та 4.4 — Інтерфейс Мессенджера

Для зручності користувачів була додана темна тема.

Якщо перейти до вкладки “Contacts” то можна додавати контакти за Id. Після того як користувач поділився своїм Id наприклад зі своїм другом, то клієнт може додати його в контакти, що зображено на рисунку 4.5

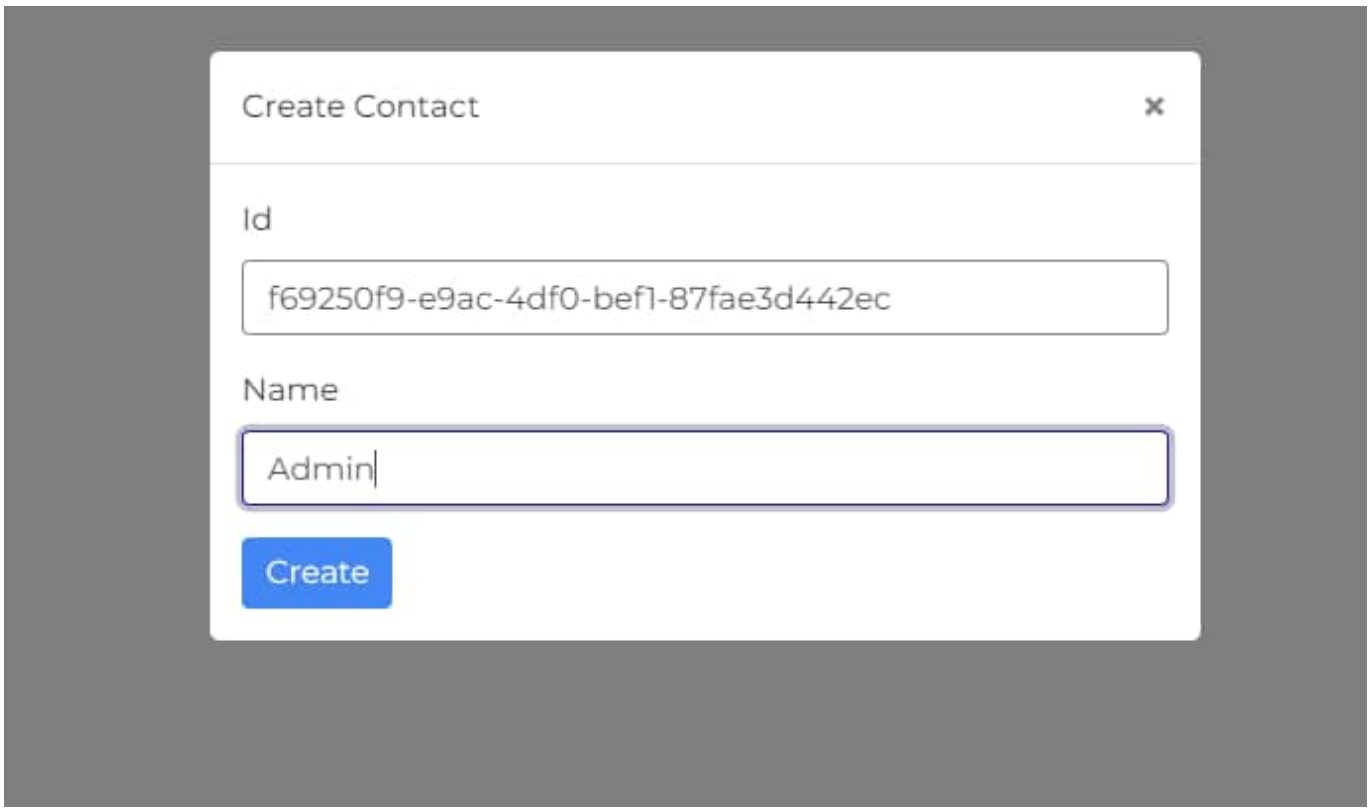


Рисунок 4.5 — Інтерфейс Мессенджера

Після додання контаку ми можемо розпочати чат к іншим користувачем, що зображенно на рисунку 4.6.

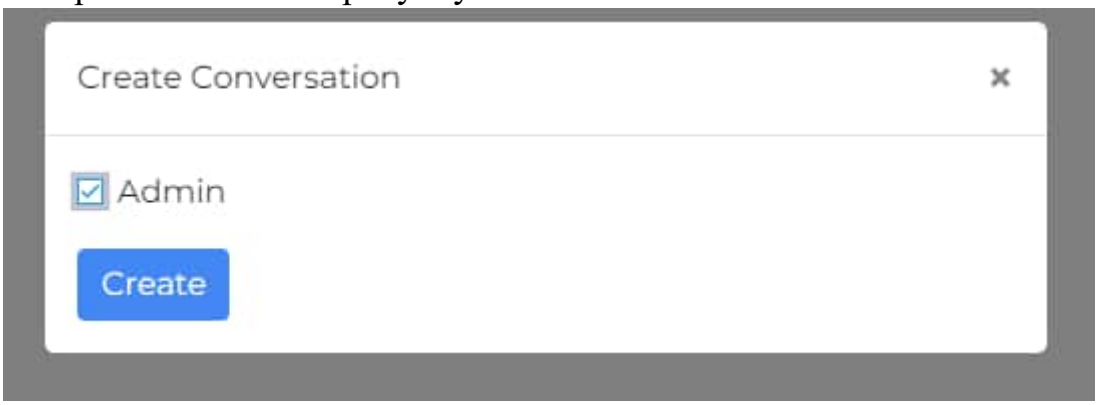


Рисунок 4.6 — Інтерфейс Мессенджера

Після чого можна розпочинати розмову:

Рисунок 4.7 — Інтерфейс Мессенджера

5. Розробка системного програмного забезпечення

5.1 Обрані інструменти для реалізації програми

Розробником була обрана інтегрована середа розробки Visual Studio Code (VS Code) для створення месенджера. VS Code є продуктом, що розроблений самою компанією Microsoft і він використовується як інтегрована середа розробки (ICP). Вона надає зручне та потужне середовище для написання коду, налагодження, керування версіями та інших розробчих завдань. Ось деякі переваги, пов'язані з використанням VS Code:

1. Кросплатформенність: VS Code підтримує операційні системи Windows, macOS і Linux, що дозволяє розробникам працювати на своєму улюбленому ОС без переходу на іншу платформу.

2. Розширюваність: VS Code має потужну систему розширень, яка дозволяє розробникам встановлювати додаткові плагіни для підтримки різних мов програмування, фреймворків, інструментів і тем оформлення. Це дозволяє налаштувати середовище розробки під власні потреби.

3. Зручний інтерфейс користувача: VS Code має простий і інтуїтивно зрозумілий інтерфейс користувача. Він має легку навігацію, можливість швидкого пошуку та заміни коду, автодоповнення, відступи, підсвічування синтаксису і багато іншого, що полегшує роботу розробників.

4. Інтеграція з Git: VS Code має вбудовану підтримку Git, що дозволяє розробникам керувати версіями свого коду, створювати гілки, робити коміти та злиття безпосередньо з редактора.

5. Налагодження: VS Code надає можливості налагодження коду з використанням різних мов програмування та фреймворків. Він підтримує точки зупину, крокування по коду, спостереження за змінними тощо, що допомагає виявляти та виправляти помилки.

6. Швидкість та продуктивність: VS Code працює досить швидко і має низьке споживання ресурсів, що дозволяє розробникам бути продуктивними без зайвих затримок.

Загалом, VS Code є потужною та високоефективною інтегрованою середою розробки, яка надає розробникам широкий набір функцій, розширюваність та зручний інтерфейс користувача.

5.2 Створення алгоритмів для програмного забезпечення, яке використовується на стороні клієнта

Додаток, що розробляється, має низку функцій, включаючи:

1. Автентифікація абонента: Користувач вводить пароль на сторінці логіну/реєстрації. Пароль перетворюється на хеш-значення за допомогою функції bcrypt.
2. Створення текстових повідомлень: Користувач може створювати текстові повідомлення та надсилати їх на сервер.
3. Шифрування та дешифрування повідомлень: Листування шифрується симетричним криптоалгоритмом за допомогою ключа AES-128. Ключ змінюється кожен раз перед відправкою повідомлення. Сам ключ також шифрується за допомогою AES-256 та відправляється разом з зашифрованим повідомленням. Повідомлення шифруються перед надсиланням на сервер і дешифруються на клієнті для відображення його змісту.

4. Захищена передача повідомлень: Повідомлення надсилаються та приймаються у захищеному вигляді, забезпечуючи конфіденційність та цілісність даних під час передачі.

5. Отримання повідомлень від учасників бесіди: Клієнт отримує зашифроване повідомлення з сервера.

Якщо клієнт залогінився, він отримує можливість надсилати іншим користувачам повідомлення.

Загальна схема шифрування та розшифрування даних та їх передачі між клієнтом та сервером така:

1. Клієнт:
   - Генерує випадковий ключ.
   - Перетворює ключ в формат UTF-8 та кодує його в шістнадцятковий рядок.
   - Шифрує ключ за допомогою секретного ключа `secretKey`, отримує зашифрований ключ.
   - Шифрує введений текст повідомлення за допомогою ключа, отримуючи зашифрований текст.
   - Відправляє отримувачам ID вибраної розмови, зашифрований текст та зашифрований ключ на сервер.

2. Сервер:
   - Отримує від клієнта отримувачів ID розмови, зашифрований текст та зашифрований ключ.
   - Відправляє отримувачам зашифрований текст та зашифрований ключ.

3. Клієнт:

- Отримує зашифрований текст та зашифрований ключ від сервера та розшифровує спочатку ключ і потом саме повідомлення за допомогою розшифрованого ключа.
- Обробляє та відображає розшифрований текст у вікні повідомлення.

5.3 Розробка програмного алгоритму на стороні сервера месенджера

Для забезпечення комунікації між сервером і клієнтом було вирішено використовувати протокол WebSocket як основу зв'язку. WebSocket є протоколом, який забезпечує постійне двостороннє з'єднання між сервером і клієнтом, дозволяючи їм обмінюватися даними в реальному часі.

WebSocket є чудовим вибором для месенджерів, оскільки він дозволяє надсилати повідомлення без зайвої затримки і без необхідності постійного встановлення нових з'єднань для кожного повідомлення. За допомогою WebSocket можна відправляти повідомлення в обидва напрямки (від сервера до клієнта і від клієнта до сервера) і забезпечувати швидку та ефективну комунікацію.

Отже, обрання WebSocket є правильним рішенням для реалізації зв'язку між сервером і клієнтом у контексті месенджера.

Сервер функціонує як посередник між користувачем іншим користувачам чату, передаючи та отримуючи зашифровані повідомлення. Більш того, сервер відповідає за отримання зашифрованих даних.

Сертифікація протоколу WSS є захистом від атаки типу «людина посередині».

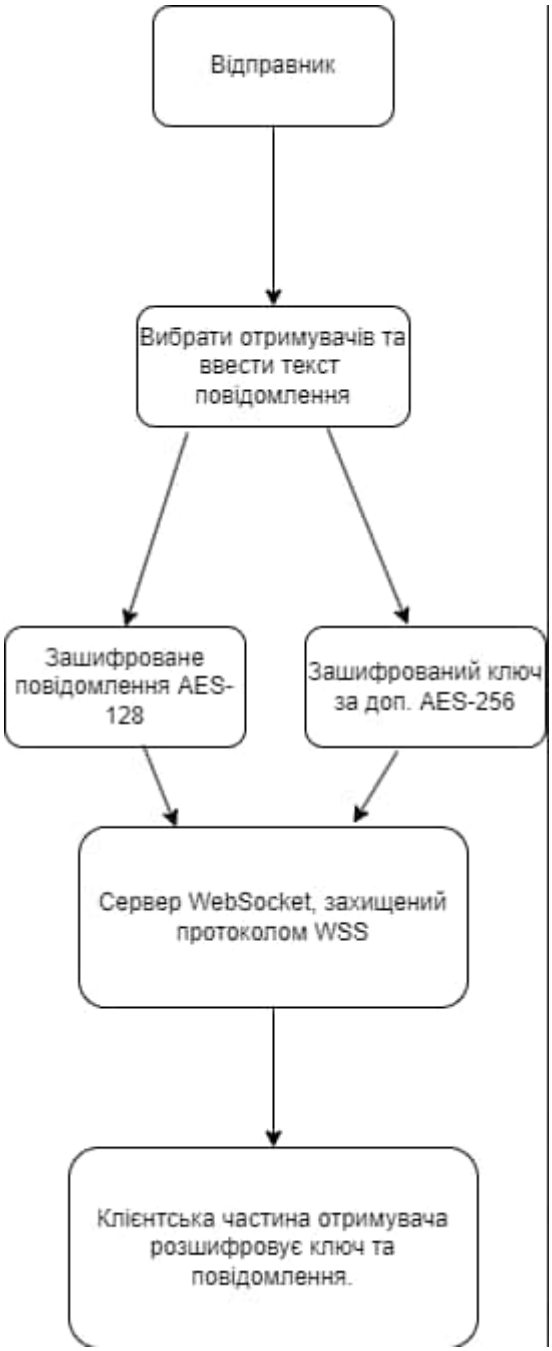


Рисунок 5.1 – Схема шифрування та розшифрування повідомлень та відправляння на сервер.

5.4 Висновок щодо розробки системного програмного забезпечення

В даному розділі розглянуто основні принципи та механізми, що забезпечують безпечний обмін повідомленнями в месенджері та збереження даних під час їх передачі через відкриті канали.

6. Як користуватися системою

6.1 Допомога з застосуванням цієї системи

Після відкриття веб застосунка клієнт бачить віконце авторизації, а якщо точніше, то логіну та реєстрації.

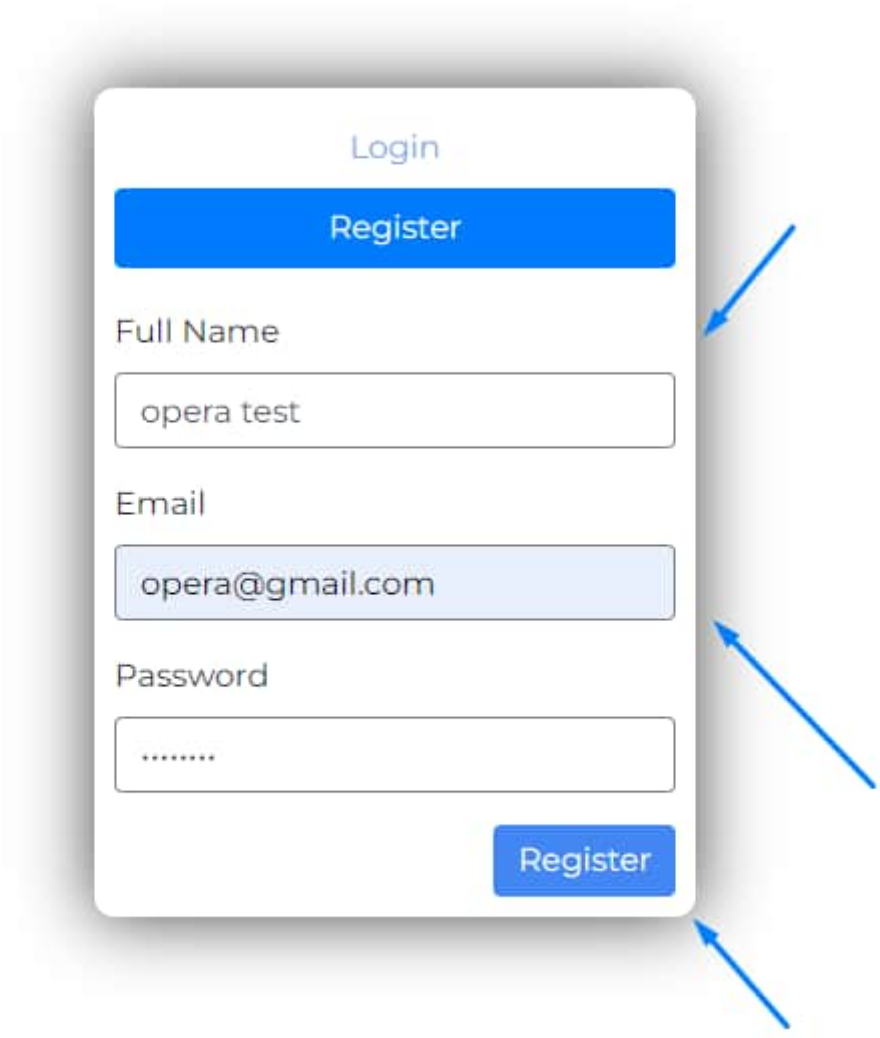


Рисунок 6.1 — Реєстрація

Після успішної реєстрації бачимо, що все добре

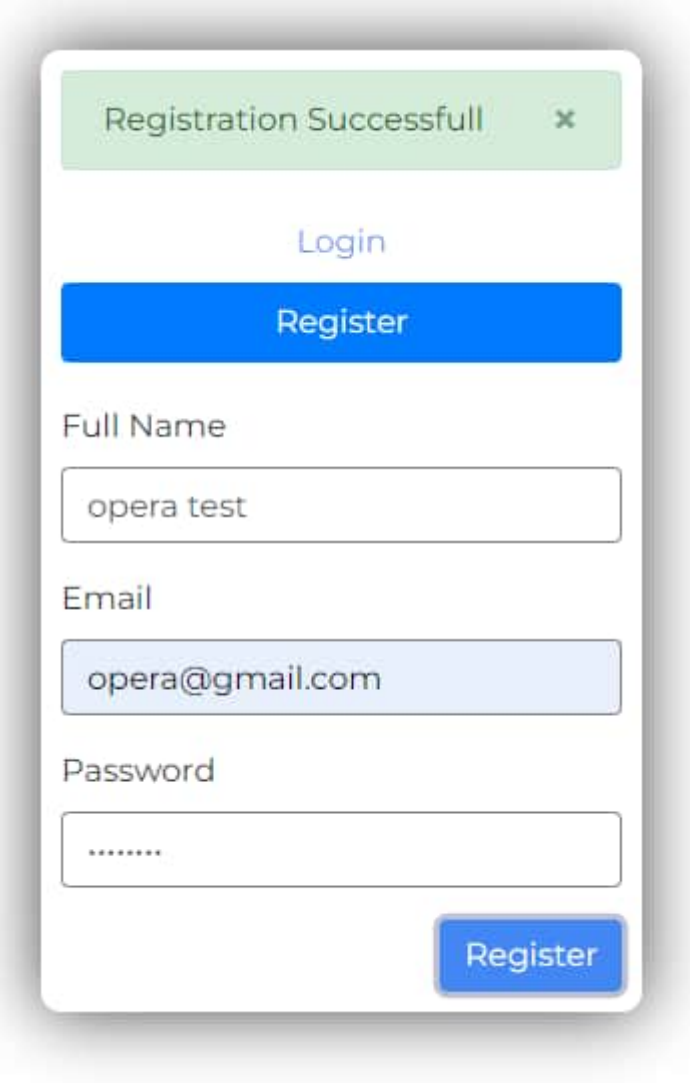


Рисунок 6.2 – Успішна реєстрація

Після чого реєстрації можемо вже пройти автентифікацію.

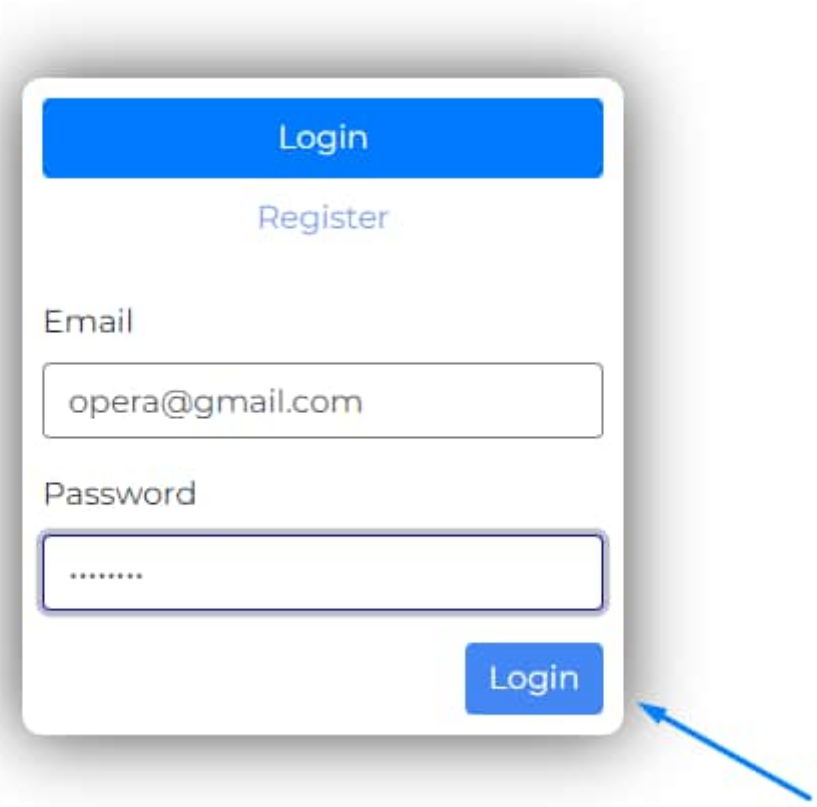


Рисунок 6.3 – Автентифікація

Коли користувач правильно ввів пароль, автентифікація буде успішною, і на екрані з'явиться головний екран. Після цього ми переходимо до розділу "Контакти" та додаємо нового співрозмовника.

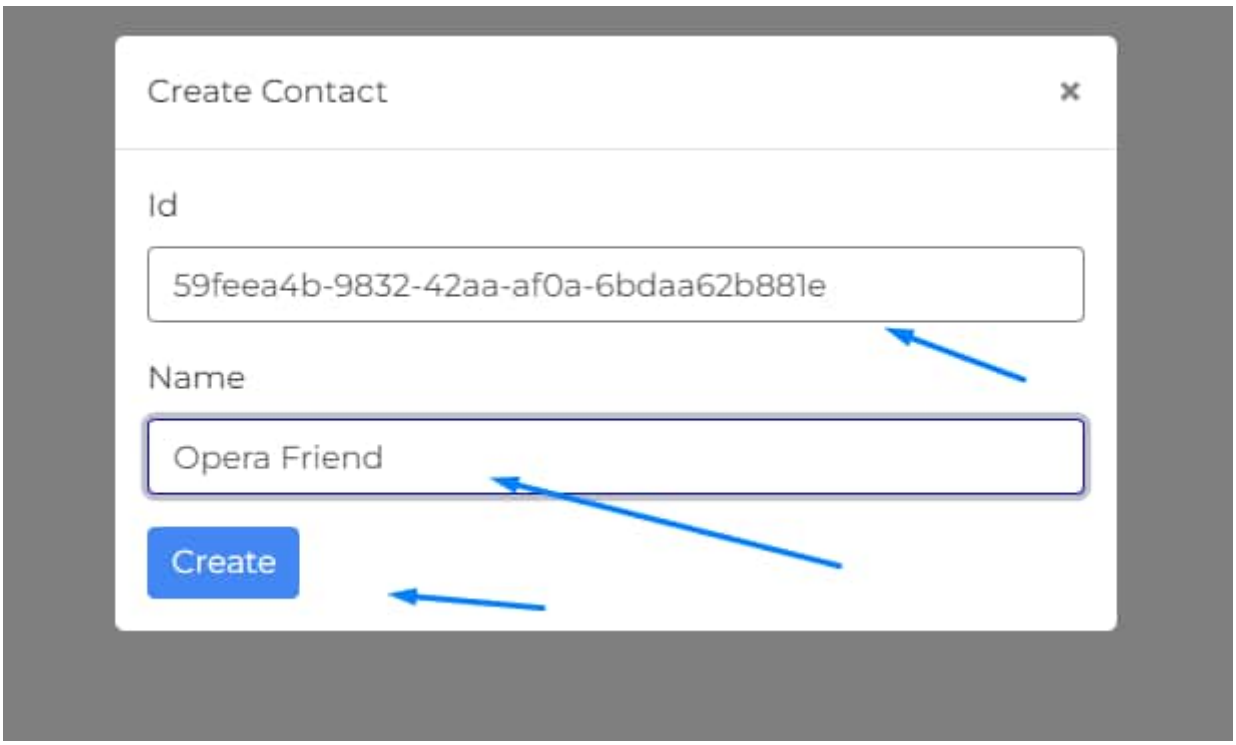
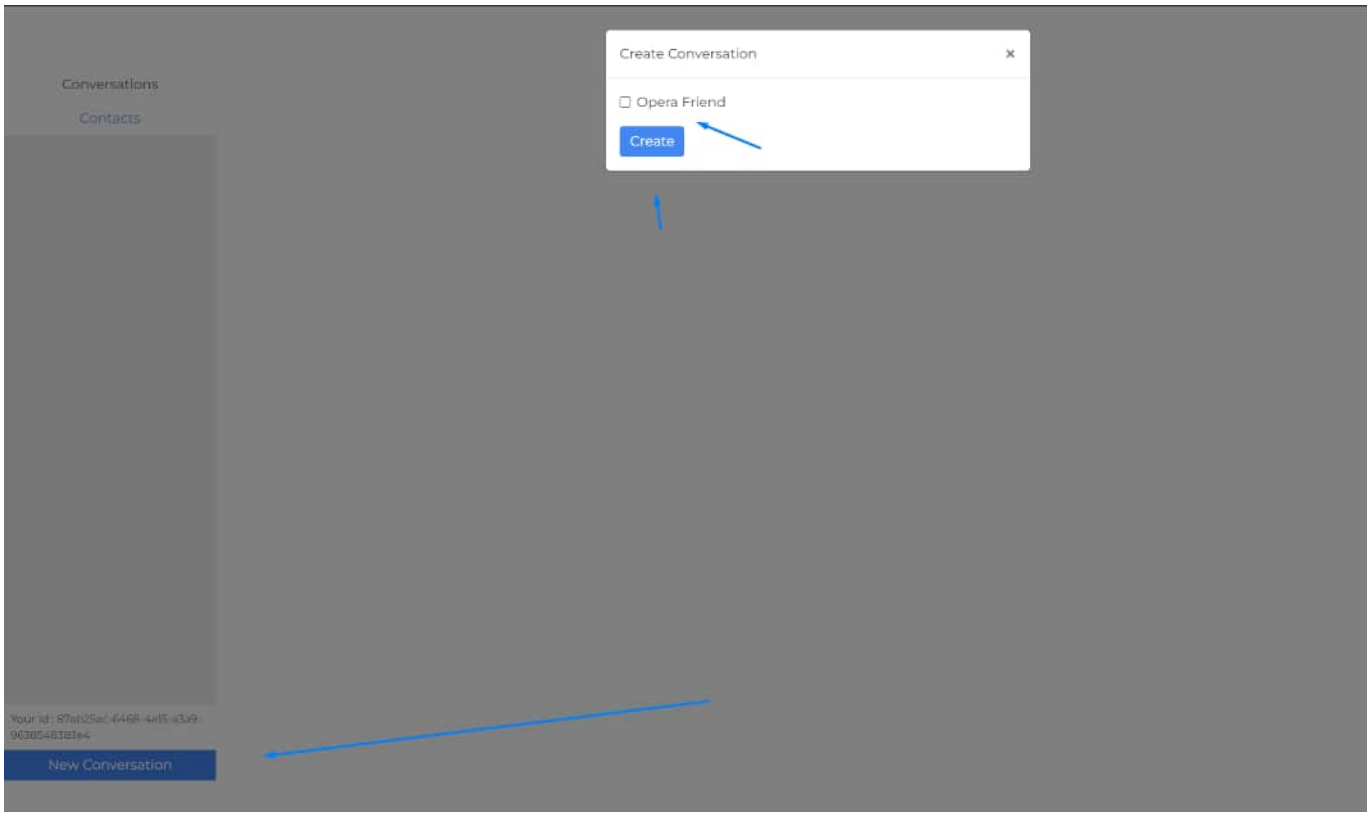
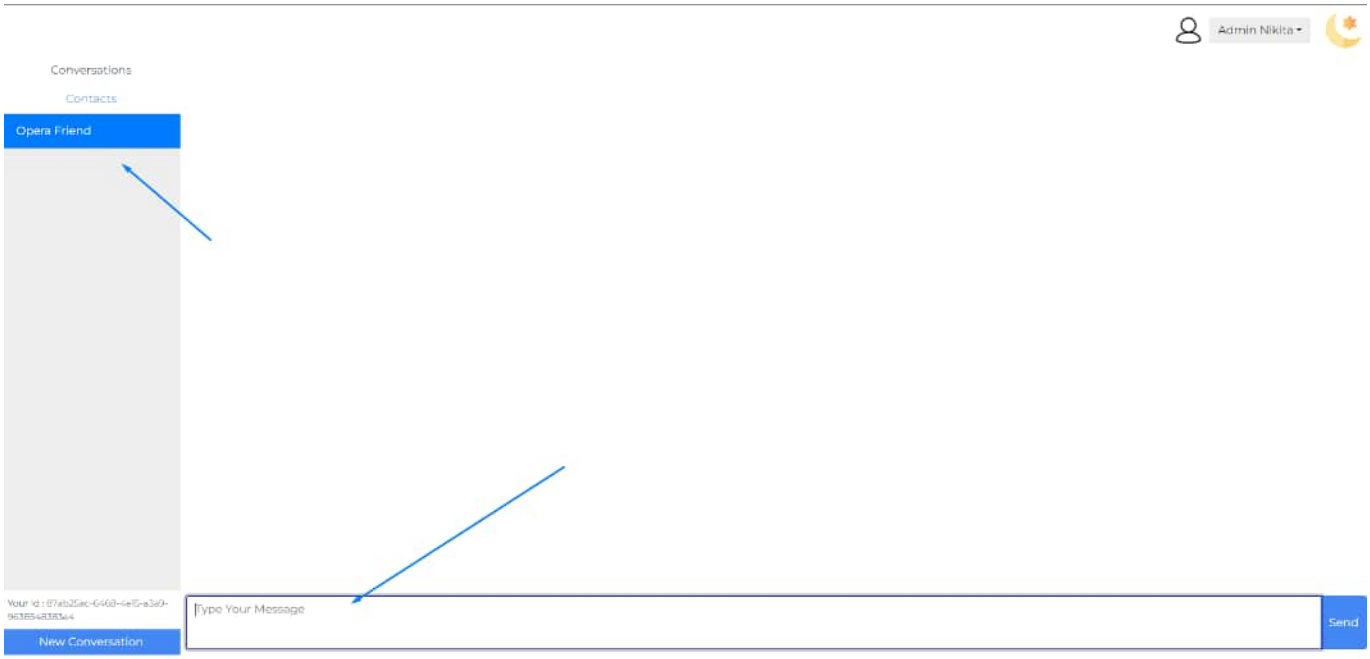


Рисунок 6.4 – Створення контакта

Після можна створити з даним контактом діалог.





Рисунки 6.5 та 6.6 – Створення діалогу з контактом

Після чого починаємо спілкування з користувачем.

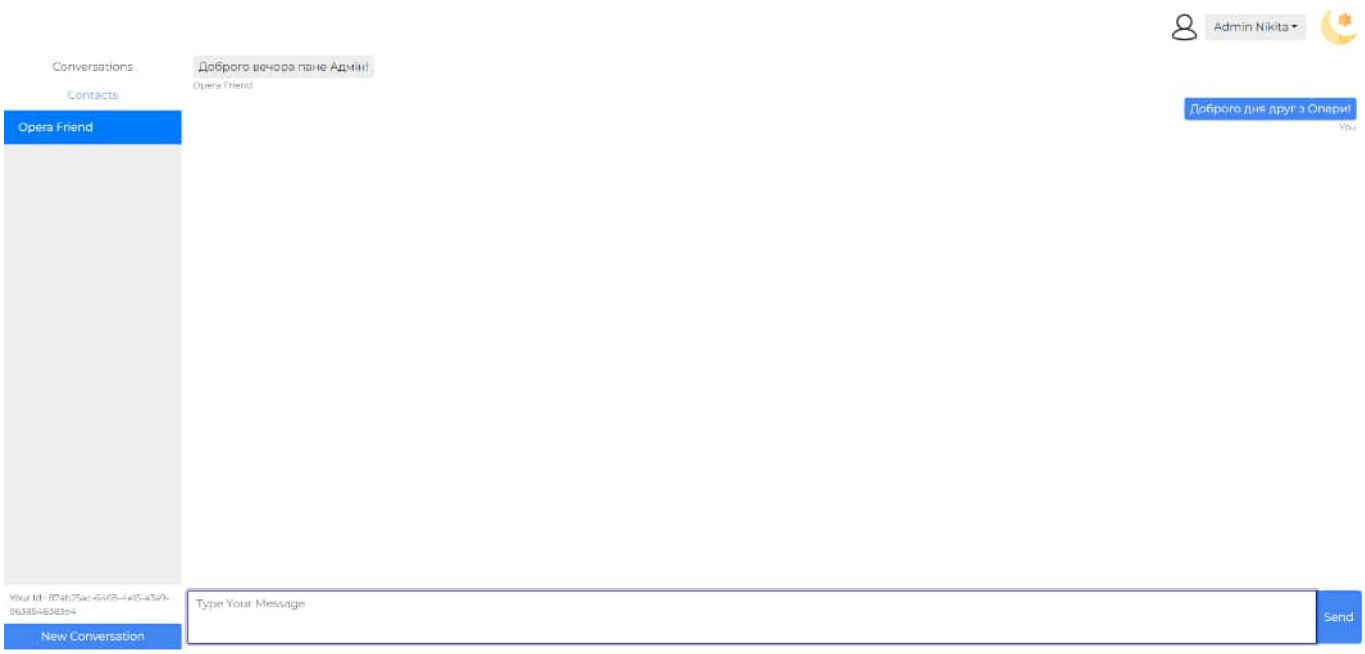


Рисунок 6.7 – Розмова з співрозмовником

ВИСНОВОК

В поточній дипломній роботі була створена веб-платформа для захищеного обміну повідомленнями, яка забезпечує високий рівень безпеки даних. При розробці системи було враховано можливість наявності шкідливого шпіонського програмного забезпечення на пристрої користувача, а також можливість перехоплення повідомлень зловмисником під час їх передачі між сервером і клієнтом.

В результаті перевірки, месенджер був визнаний повністю працездатним без виявлення жодних несправностей.

Розроблена система також рекомендується для використання в навчальному процесі як демонстраційний комплекс для забезпеченого елементарного інформаційного зв'язку між абонентами на основі клієнт-серверної архітектури.

ПЕРЕЛІК ПОСИЛАНЬ

1. React Documentation. [Електронний ресурс]. - Режим доступу: https://legacy.reactjs.org/docs/react-component.html
2. Telegram Encryption. [Електронний ресурс]. - Режим доступу: https://core.telegram.org/api/end-to-end
3. Telegram API Documentation. [Електронний ресурс]. - Режим доступу: https://core.telegram.org/api/
4. Viber API Documentation. [Електронний ресурс]. - Режим доступу: https://developers.viber.com/docs/api/rest-bot-api/
5. End-to-End Viber Encryption. [Електронний ресурс]. - Режим доступу: https://help.viber.com/hc/en-us/articles/8909167863453-End-to-End-Encryption-in-Chats#:~:text=How%20does%20end%2Dto%2Dend,they%20reach%20the%20intended%20recipient.
6. Securing Web Transactions, TLS Server Certificate Management. [Електронний ресурс]. - Режим доступу: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-16.pdf
7. Computer Networking and Internet Protocols. [Електронний ресурс]. - Режим доступу: https://www.cse.wustl.edu/~jain/tutorials/ftp/airtel.pdf
8. WebSocket(Socket.IO) Documentation. [Електронний ресурс]. - Режим доступу: https://socket.io/docs/v4/
9. RSA Encryption. [Електронний ресурс]. - Режим доступу: https://www.amsi.org.au/teacher_modules/pdfs/Maths_delivers/Encryption5.pdf
10. Diffie-Hellman Encryption. [Електронний ресурс]. - Режим доступу: http://docenti.ing.unipi.it/g.dini/Teaching/sncs/lectures/handouts/04.Diffie-Hellman.pdf
11. AES Encryption. [Електронний ресурс]. - Режим доступу: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf
12. Crypto Hash Functions. [Електронний ресурс]. - Режим доступу: https://www.cse.wustl.edu/~jain/cse571-11/ftp/l_11chf.pdf
13. Bcrypt documentation. [Електронний ресурс]. - Режим доступу: https://www.npmjs.com/package/bcrypt
14. AES understanding. [Електронний ресурс]. - Режим доступу: https://www.appsealing.com/aes-128-encryption/