**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА**

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від ___ червня 2024 року, протокол № ___.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА
на тему:
# «РОЗРОБКА ОНЛАЙН ЧАТУ З ВАРІАТИВНИМ ШИФРУВАННЯМ»

**Виконав:**
студент 4-го курсу
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОПП «Інформаційна безпека телекомунікаційних систем і мереж»
Жулінський Євгеній Костянтинович _____

**Науковий керівник**:
канд. фіз.-мат. наук, доцент
Кононов Михайло Володимирович _____

**Рецензент:**
канд. фіз.-мат. наук, доцент
Єфіменко Світлана Володимирівна_____

Засвідчую, що у цій бакалаврській роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент Жулінський Євгеній Костянтинович_____

# РЕФЕРАТ

Дипломна робота: с. 50, табл. 20, рис. 44, джерел 12.

ОНЛАЙН ЧАТ, PHP, AJAX, JAVA SCRIPT, MySQL.

Мета роботи – розробка веб-чату з варіативним шифруванням.

Розроблено онлайн чат – месенджер. Використано методи криптографічного захисту, що дозволяють підвищити рівень безпеки обміну повідомленнями.
Замість єдиного алгоритму шифрування , який може бути вразливим до зламів, варіативний підхід використовує кілька алгоритмів та можливість зміни параметрів в процесі роботи. Для шифрування було використано алгоритми: DES, AES, RSA, El-Gamal, а для шифрування ключів в свою чергу було використано Diffie-Hellman.

Використано зв'язку PHP і AJAX – потужний інструмент для створення динамічних та інтерактивних веб-сайтів. Це забезпечує асинхронність, адже AJAX дозволяє взаємодіяти з сервером без необхідності перезавантаження сторінки. Це дозволяє створювати більш швидкі та ефективні веб додатки, оскільки можна завантажувати лише частину сторінки, яка потребує оновлення.

Внаслідок використання стратегії оптимізації та сегментування коду було суттєво зменшено кількість об'єднаних складових, також з'явилися можливості для швидкого й зрозумілого рефакторингу в дизайнерскому та функціональному планах.

# ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

DES – data encryption standard

AES – advanced encryption standard

XOR – exclusive OR

HEX – шістнадцяткова система числення

RSA – Rivest-Shamir-Adleman алгоритм

D/F – Diffie-Hellman алгоритм

El/G – El-Gamal алгоритм

E2EE – end to end шифрування

SSL – Secure socket layer

TLS - transport layer security

# ВСТУП

Онлайн чатинг зародився на зорі інтернету – це були просто два поля на сайті. В одному відображався текст, що набирається, в іншому вже написаний. В просунутих варіантах історія повідомлень зберігалася в базах даних, на більшості ж ресурсів вмістилищем інформації слугував звичайний текстовий файл.

Звісно такий спосіб зберігання не мав ніякого ступеня захищеності. Конфіденційність інформації гарантувалася лише чесністю та надійність людини, або корпорації, що тримала базу даних. Цим почали користуватися зловмисники. Тож розробники почали презентувати рішення з шифруванням даних.

В 2001 році був представлений AES, це стало проривом в кібербезпеці. Він і досі є одним з найпопулярніших та захищених алгоритмів шифрування інформації. Далі почався розквіт комп'ютерної криптографії. Одні математики й програмісти винаходили свої, інші ж доводили їх нежиттєздатність знаходячи вразливість. Так з'явилося багато алгоритмів та протоколів кодування різних сигналів.

Проте саме 5 алгоритмів винайдених до 2001 року наразі являють собою криптографічний базис месенджерів та чатових додатків. DES, AES, RSA, D/F, El/G. Причому перші два використовуються для всього потоку інформації, а інші для створення та шифрування ключів.

Багато компаній розгортають свої застосунки на основі дешевих та легких до налаштування, готових компонентів, що не мають відповідного ступеню захищеності й покладаються на надійність бази даних. Для деяких така безпрецедентна зухвалість відносно цього питання коштує дуже багато. Неконтрольований виток інформації про новий продукт, атака на БД й використання інформації проти компанії чи окремих особистостей, тисячі інших випадків? Все це є факторами ризику, тож нехтування правилами кібербезпеки є першим етапом краху серйозної структури.

# 1. ЗАСОБИ ОБМІНУ ПОВІДОМЛЕННЯМИ

## 1.1. Порівняння функціональності різних платформ обміну повідомленнями

Веб-чати стали важливою частиною сучасного цифрового спілкування. Від перших текстових повідомлень до сучасних багатофункціональних платформ – вони значно еволюціонували. Розглянемо історичний розвиток веб-чатів[**1**] та сучасні методи шифрування, що забезпечують високий рівень конфіденційності користувачів.

Перші форми онлайн-чатів з’явилися у 1980-х роках з розвитком інтернету. Однією з перших і найбільш впливових платформ був Internet Relay Chat (IRC), створений Ярко Ойкаріненом у 1988 році у Фінляндії. IRC дозволяв користувачам спілкуватися один з одним за допомогою текстових повідомлень у різних каналах. Ця платформа стала популярною серед користувачів комп’ютерів у всьому світі, що стимулювало розвиток подібних сервісів.

Дослідники й фахівці в цій області відзначають, що з поширенням доступу до інтернету у людей значно зросло зацікавлення в спілкуванні через-онлайн платформи, це ще більше спонукало розвиток схожих сервісів, зокрема платформи IRC.

У 1990-х роках з’явилися одні з найпопулярніших чат-платформ -вони стали символами свого часу. AOL Instant Messenger (AIM), запущений у 1997 році, став першою популярною чат-платформою в Сполучених Штатах, відкриваючи нові можливості для віддаленого спілкування. ICQ - був випущений у 1996 році, також став одним із перших, найпопулярніших месенджерів, що підтримував обмін повідомленнями між користувачами через інтернет. Ці платформи відіграли ключову роль у поширенні культури онлайн-спілкування та встановленні нових стандартів в цій галузі.

На початку 2000-х років разом з шаленим розвитком веб-технологій з’явилися графічні інтерфейси[**2**] для чатів, що робило їх більш доступними для широкої аудиторії. Це призвело до збільшення популярності платформ та стимулювало їх подальший розвиток. Соціальні мережі, такі як Facebook, швидко зрозуміли, що спілкування у реальному часі для їх користувачів є важливим пунктом і почали інтегрувати чат-функції безпосередньо у свої платформи. Це дозволило користувачам обмінюватися повідомленнями з друзями та контактами безпосередньо на сторінках соціальної мережі. Прикладом такої інтеграції є Facebook Chat, згодом став відомим, як Facebook Messenger.

У сучасному світі існує велика кількість різноманітних чат-платформ, що відповідають на різноманітним потребам користувачів. Найбільш популярними серед них є WhatsApp, Facebook Messenger, Telegram, Slack, Microsoft Teams та багато інших. В дослідженні «Найпопулярніші месенджери світу»(2024)[**3**], Фабіо Дуарте навів дані про кількісне порівняння застосувань того чи іншого месенджера у всьому світі. Дані наведені за 2020-2021 рік, на рисунку 1.1.

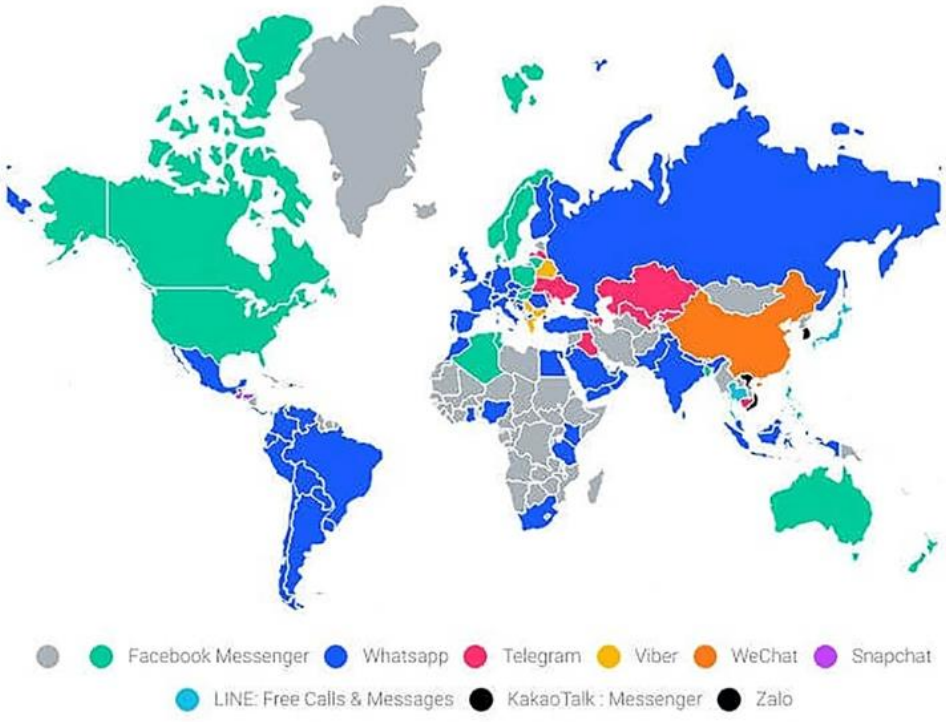


Рисунок 1.1 – Територіальна популярність.

## 1.2. Інформаційна безпека в засобах обміну повідомленнями

Наразі телекомунікації мають просто шалені темпи розвитку. Цифрові технології стали частиною нашого життя, що має доступ до особистої інформації. Основним способом обміну інформацією стали месенджери. Текстові повідомлення, відео, аудіозв'язок, файловий обмін, геодані та інші типи конфіденційної інформації збираються месенджерами та зберігаються в базах даних. Звичайна людина знає дуже мало про приватність та конфіденційність. Не розуміє різниці між пропрієтарним програмним забезпеченням та програмами з відкритим кодом. Нижче проаналізовано основні питання приватності.

Пропрієтарне програмне забезпечення – те ПЗ, що має власника, або автора, якому належать всі права програму. Власник має право вносити абосолютно будь-які зміни , чи ж встановлювати абонентський збір за використання застосунку. В більшості випадків власник накладає правила користування і абонент повинен бути з ними згоден, щоб мати можливість користуватися програмою.

Вільне програмне забезпечення  надає будь-кому дозвіл на модифікацію, розповсюдження та користування програмою, також таке користування не обмежене правилами. Розробниками такого ПЗ часто виступає група ідейних осіб, що мали потребу й організували продакшн. Всі рішення по кінцевій модифікації базового коду приймаються колективно, тож в такому випадку відсутня монополія. Також відкритий код дає змогу незалежним експертам проводити тестування та пошук вразливостей використовуючи будь-які методи.

Алгоритми шифрування наразі є однією з найвужчих частин в аспекті контролю такого програмного забезпечення. В більшості випадків месенджери працюють так, що повідомлення шифрується в серверній частині, цей варіант є абсолютно ненадійним. Але в сучасних програмах, що претендують на конфіденційність, використовується модель E2EE.

E2EE – це наскрізне шифрування, що оброблює інформацію на стороні адресанта, та дешифрує її на стороні адресата. Має сенс використання, як

симетричних, так і асиметричних алгоритмів шифрування. Повідомлення ж зберігаються на серверах, або користувацьких пристроях виключно у зашифрованому вигляді – неможливому для декодування.

Є варіант розташування інформації за допомогою протоколу XMPP. Це протокол передачі, був представлений у 1999 році. В базу методу укладено протокол XML. Його ідеєю - це використання децентралізованої системи розташування серверів, де користувач може розгорнути свій сервер і стати одним з вузлів, або ж просто приєднатися до вже існуючого. Проте не так багато людей готові стати вузлом, а розгортати купу серверів коштує дорожче ніж зробити централізовану систему, а також вбиває сенс, бо всі сервери будуть належати одній компанії, що означає монополію. Тож цей метод не набув широкого застосування.

Існували питання ідентифікації користувача в системах, де не зберігається велика кількість метаданих. Більшість месенджерів їх збирають, тож у кожного користувача є свій ID. Системи, що пропонуються, як абсолютно децентралізовані стикнулися з проблемою підтвердження особистості людини з якою користувач має діалог. З цим призвана боротися бібліотека функцій PGP, що в даний час є найкращим рішенням шифрування цифрових підписів даних. Створена у 1991 році. В країнах, що мають електронні кабінети громадян використовується саме ця технологія, проте вона не вберігає бази даних.

Для шифрування даних, що надсилаються використовують протоколи:

Signal Protocol – презентований Open Whisper Systems (OWS). Сьогодні його використовує Signal. Також використовувався WhatsApp, Viber, Facebook Messenger, проте вони перейшли на інші протоколи, або ж заснували свої на основі. Асинхронний протокол, що робить повідомлення абсолютно незв’язними, забезпечує нормальний рівень узгодженості користувачів. Являє собою зв’язку алгоритмів: Axolotl, 3-DH, SHA-256, AES-256, Curve25519.

Matrix – метод федеративного типу, що працює на JSON та концептуально нагадує XMPP. Matrix має купу власних вузлів, а також надає можливості для створення власних. Наразі проявляє шалений попит в стартапах через наявність створеної бази серверів, а також можливість інтеграції в інші платформи.

AES – золотий стандарт шифрування, розроблений та затверджений в США Має в основі симетричний ключ, це значить, що один ключ використовується і для шифрування і для дешифрування даних. Являє собою просунуту версію DES.

SHA – криптографічне хешування. Відповідно до вимог повинен використовуватися усіма державними корпораціями у Сполучених Штатах. Розроблений агентством національної безпеки США в 1993 році. Після компрометизації першої версії алгоритму було SHA-2 та SHA-3.

MD5 – 128-бітне хешування. Розроблений в MIT у 1991 році. Використовується для хешування паролів та являє собою значно посилену версію MD4. Має офіційні вразливості RFC 6151, її планують прибрати до 2027 року.

ECDH – алгоритм Diffie-Hellman, розроблений в США М. Хелманом та У. Діффаєм. Дозволяє всім користувачам отримати один ключ використовуючи компрометований канал зв’язку. Саме цей ключ потім і використовують для симетричного шифрування та дешифрування інформації. Цей винахід став проривною технологією в світі кібербезпеки, адже викорінив своїм появленням проблему розподілу ключів між користувачами.

Salsa20 – шифрування розроблене Д. Бернштейном, вперше представлений на виставці алгоритмів шифрування. Він отримав перемогу в категорії шифрів для великих масивів інформації. Математично нагадує AES, проте використовується хеш-функція на 20 циклів. Має вразливість до атак за часом, тож в оновленій версії XSalsa20 введено імунітет до такої вразливості.

Axolotl – алгоритм управління створеними ключами шифрування. Створений засновником Signal в 2013 році. За основу взята хеш-функція та протокол ECDH Використовує проривну функцію, яка у разі компрометизації повністю блокує зловмиснику доступ до відкритої частини інформації.

Poly 1305 – хеші, що використовуються в шифруванні як одноразовий код для автентифікації що має на увазі спільний ключ для використання між адресантом та адресатом повідомлення.

Curve 25519 – метод, що робить можливим при 256-бітному розмірі ключа створити 128-бітне шифрування. Використовується в методі Diffie-Hellman, є найшвидшим методом, що на додачу позиціонується, як Open Source розробка. Також відомий під навою X 25519 і застосовується в базових мережевих протоколах, наприклад OpenSSL.

PFS – принцип криптування, що робить фактично неможливим компрометацію даних в разі розшифровки будь-якого з ключів. Його назва «Досконала секретність» вказує на рівень надаваної безпеки.

DHT – є децентралізованими хеш-функціями, що розподілені завдяки федеративному устрою. Фактично об’єднує всі сервери, хости. Висока відмовостійкість, масштабованість та легкість в організації є перевагами при використанні. Дуже часто використовується в децентралізованих файло-обмінниках з нелегальними та неліцензійними даними.

OMEMO – протокол, що застосовується в XMPP. Має можливість асинхронної передачі повідомлень. Має на борту алгоритм, що забезпечує шифрування від кількості, цим забезпечує гарну синхронність повідомлень між клієнтами незалежно від стану підключення до мережі.

OTR – протокол для обміну повідомленнями між двома користувачами. Використовується в peer-to-peer мережах. Включає в себе алгоритми SHA-1, Diffie-Hellman, симетричний ключ взято від AES. Головною метою при створенні було забезпечення можливості спілкування без будь-яких записів про нього.

Всі ці принципи шифрування є значними досягненнями криптографії. Вони перевірялися купою тестів і навіть відповідають принципу Керкгофза. Він каже, що стійкість криптографічного алгоритму не має залежати від його архітектури, а лише від ключів. Тобто при оцінці якості шифрування вважається, що зловмисник знає все про її архітектуру й математичні функції, в нього під рукою всі наукові досягнення для декриптингу. І єдине чого він не знає – це ключів шифрування.

Шифрування повідомлень убезпечує нас від розкриття прямого контексту переписки, проте це лиш маленький шматок інформації про людину.
«Metadata absolutely tells you everything about somebody`s life. If you have enough metadata, you don't really need content», що перекладається на кшталт: «Метадані абсолютно все розповідають про чиєсь життя. Якщо їх у вас достатньо, вам навіть не потрібен контент». Це сказав Стюарт Бейкер, від був генеральним радником АНБ США. Будь-який централізований месенджер збирає метадані. Вони потрібні для вдосконалення власних сервісів, навчання й побудови просунутих нейронних інтерфейсів. Монетизація метаданих теж є, продаж аналітичних даних і гарячих баз комерційним структурам взагалі є найбільшим джерелом доходу тримачів месенджерів за процентним співвідношенням. І звісно співпраця з державними структурами в області безпеки – спроби відстежити та запобігти злочинам, що можуть бути загрозою суспільній, національній та міжнародній безпеці.

Всі ці дані збираються безкоштовними, централізованими системами, що фактично не надають ніяких гарантій користувачу на використання продукту. Власник думає у комерційних інтересах і це завуальовано в правилах користування, тому проблеми та витоки інформації при використанні лягають на плечі самих користувачів.

Нижче на малюнках 1.2, 1.3, 1.4, 1.5 наочно показано кількість метаданих, що збирають різні месенджери. В добірку втрапили: Signal, iMessage, WhatsApp, Facebook Messenger . Зверху назва месенджера, нижче в рамці метадані та інформація, яку він збирає та зберігає на серверах.


Рисунок 1.2 – Метадані Signal.

Як бачимо месенджер не збирає ніякої інформації, окрім базової. Про дату реєстрації та дату останньої активності акаунта. Цю інформацію зобов’язані збирати всі месенджери через судові справи щодо зникнення чи викрадення людей.


Рисунок 1.3 – Метадані iMessage.

iMessage є стандартним чатовим застосунком пристроїв на базі IOS.
Він з’явився в базовій версії IOS 5 і призваний відправляти текстові та мультимедійні файли в обхід стандартної SMS/MMS системи. Сервіс авторизації проходить завдяки особистим даним, що прив’язані до конкретного пристрою.


Рисунок 1.4 – Метадані WhatsApp.

Як бачимо меседжер, що є найбільш популярним більш ніж в половині світу окрім стандартної інформації зберігає ще й таку. Причому ви побачите всі ці пункти, читаючи правила «Угоди користувача», проте користувач, що не розуміється на кібербезпеці навіть не зверне на них увагу.

Месенджер WhatsApp вважається доволі захищеним серед більшості населення, як наприклад і Viber, що зберігає таку ж інформацію.

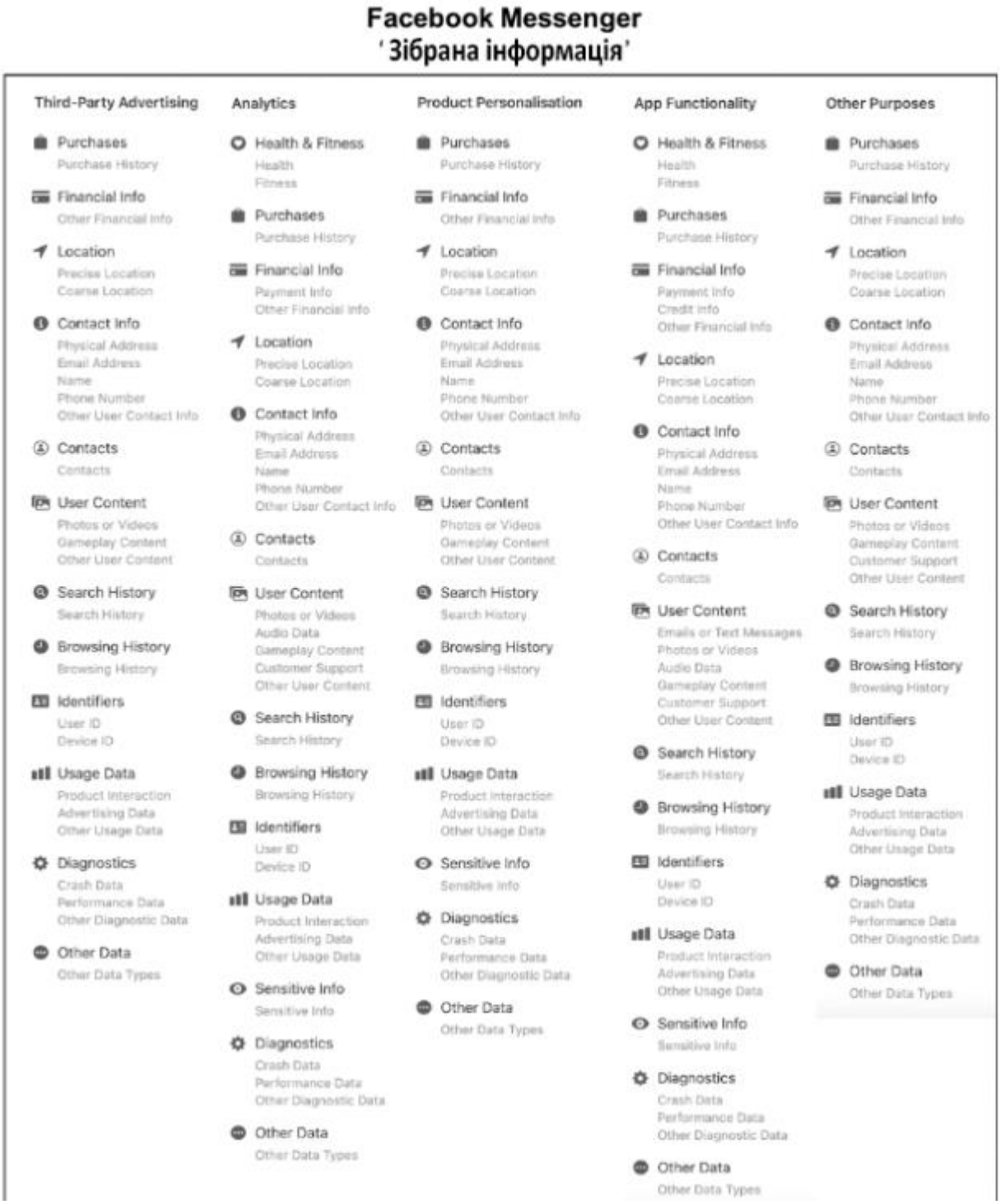


Рисунок 1.5 – Метадані Facebook Messenger.

Беззаперечним лідером по збору особистої інформації є Facebook Messenger, що належить американському технологічному гіганту Meta Platforms. Маючи розвинуту архітектуру зовнішніх додатків, вся інформація з застосунків корпорації надходить та зберігається на серверах.

Важливим фактором при виборі конфіденційного месенджера є спосіб реєстрації акаунту. В дійсно захищених осередках, що не збирають зайві метадані це реалізовано без зазначення електронної пошти та номеру телефону. Якщо дійсно конфіденційну електронну пошту можна створити використовуючи сервіси по типу Protonmail.com, то з номером телефону так не вийде – доведеться вказати свій. До прикладу знаючи ваш номер телефону, досить легко провести розвідку за класифікацією OSINT, що заснована на відкритих даних, або ж просто купити інформацію по номеру на чорному ринку. У звіті TrendMicro[**4**] наведено перевірені пункти, по яким можна пробити номер телефону. На рисунку 1.6.

| Cellular services | Starting Prices (Rubles) |
|---|---|
| Retrieve the phone number give passport information | 700 – 5,000 (US$10 to US$69) depending on cellular provider |
| Phone call and SMS records without cell tower locations | 2,000 – 35,000 (US$28 to US$482 USD) |
| Phone call records with cell tower locations | > 60,000 (US$826) |
| Blocking phone numbers | 4,000 (US$55) |
| Map where calls were located | > 10,000 (US$138) |
| Location of phone/SIM card | price depends on region |
| Printout of an SMS message | price on request |

Рисунок 1.6 – Послуги й ціни на чорному ринку.

Тобто SIM-карта є маячком для злодіїв, що потребує подальшої розробки. Важливим фактором є вразливість SS7 – зафіксована в сотових мережах і дає можливість імітувати базові станції на шляху абонента й перехоплювати дані. Таким чином перехоплюється автентифікація за номером з SMS- повідомленя, після запиту на зміну пароля.

Саме тому не рекомендую використовувати особистий номер телефону для реєстрації в онлайн сервісах.

# 2. РОЗРОБКА ВЕБ-ЗАСОБУ ОБМІНУ ТЕСТОВИМИ ПОВІДОМЛЕННЯМИ

## 2.1. Розробка архітектури

«red Chat» є веб-програмою, що забезпечує захищене спілкування в чаті за допомогою наскрізного шифрування.

Включено чотири алгоритми шифрування та один алгоритм для обміну ключами. Всі алгоритми реалізовано з нуля без використання готових бібліотек.

Алгоритми розроблені з використанням PHP та JavaScript, а потім в зашифрованому вигляді передаються в захищену базу даних MySQL за допомогою такої зв'язки технологій, як PHP та AJAX.

Користувачі повинні зареєструватися з унікальним ім'ям та паролем. Без використання електронної пошти, або номеру телефону.

Користувачі можуть відправляти текстові повідомлення та файли. Після відправки файли можуть бути завантажені обома користувачами.

Існує список вибору, що дозволяє користувачу обрати бажаний варіант шифрування повідомлення, що наразі вже надруковане, або буде надруковане в текстовому полі. Можливими алгоритмами для вибору є DES, AES, RSA, El-Gamal.

Всі файли шифруються методом шифрування їх масиву Base64, а не методом шифрування поточного шляху. Це значить, що перед шифруванням всі дані перекодовуються у формат Base64[**5**], тобто формат для представлення будь-яких двійкових даних в текстовій формі. І потім вже шифрування відбувається за рахунок обраного методу.

В списку користувачів можна дізнатися теперішній статус користувача (online/offline) за допомогою маркерів.

Окрім стандартних даних по типу дати реєстрації, дати останнього входу в базі даних зберігається інформація про час надсилання повідомлення в залежності

від використаного алгоритму шифрування. Також заходиться повідомлення в шифрованому вигляді. Чат не збирає іншої конфіденційної інформації.

Нижче на рисунку 2.1, за допомогою sequence diagram показана логіка механізму реєстрації, авторизації та запитів на оновлення статусів користувачів.

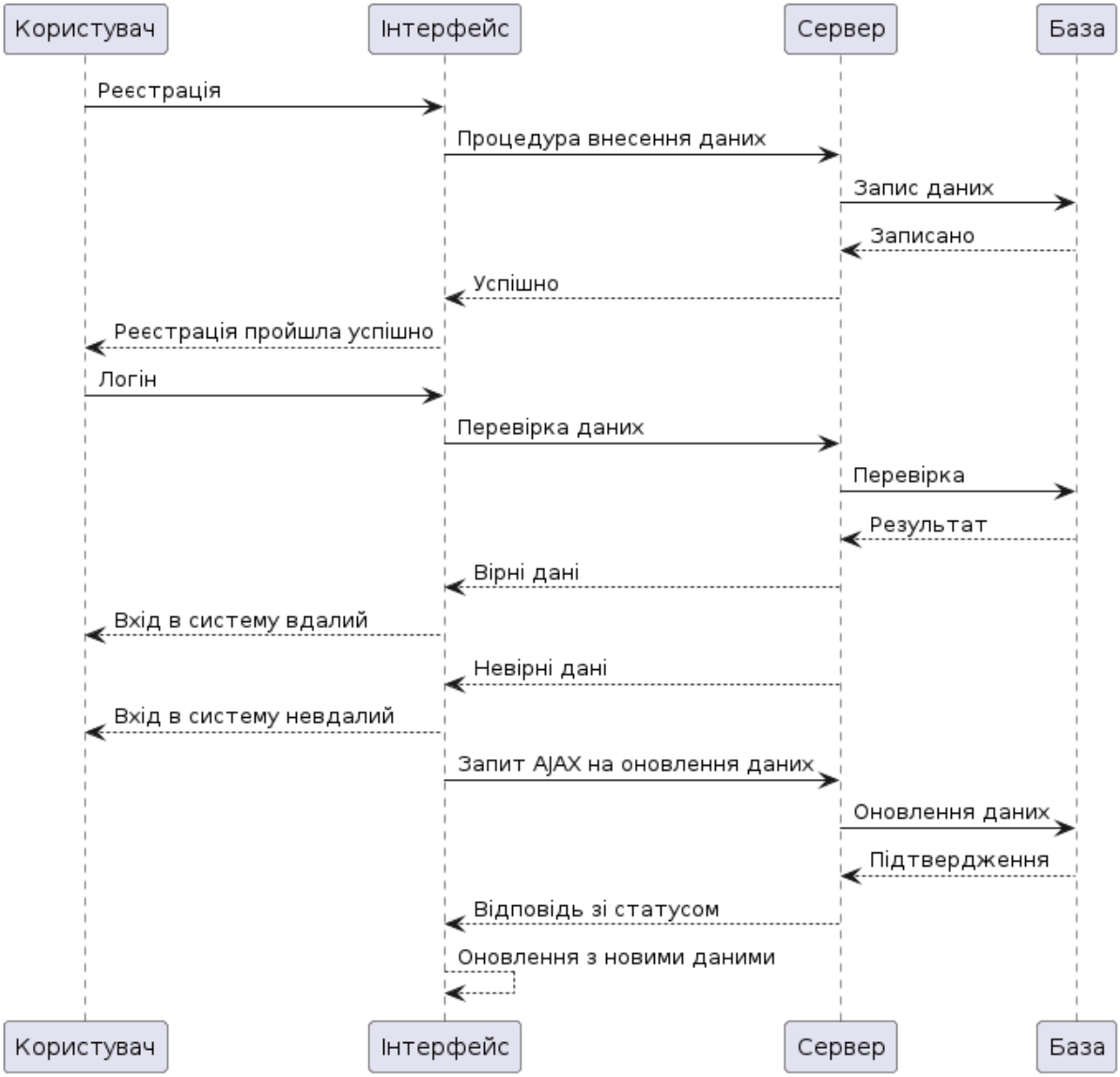


Рисунок 2.1 – Логіка роботи реєстрації, авторизації та запитів на список користувачів.

Нижче, на рисунку 2.2 буде показано логіку роботи месенджера у написанні повідомлення, вибору методу шифрування, кодування повідомлення, або файлу в формат Base64, шифрування та відправку самого повідомлення.

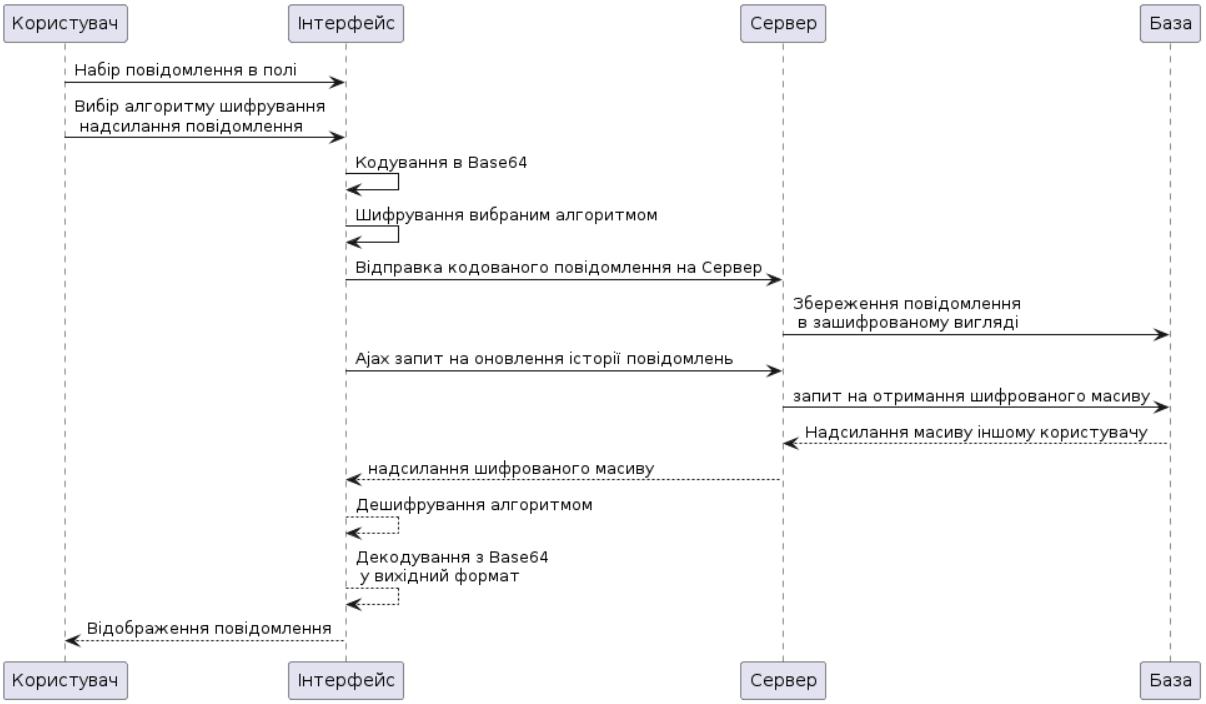


Рисунок 2.2 – Sequence diagram-а логіки роботи шифрування, відправлення, дешифрування та отримання повідомлення.

На рисунку 2.3 можна побачити конструкцію бази даних MySQL.

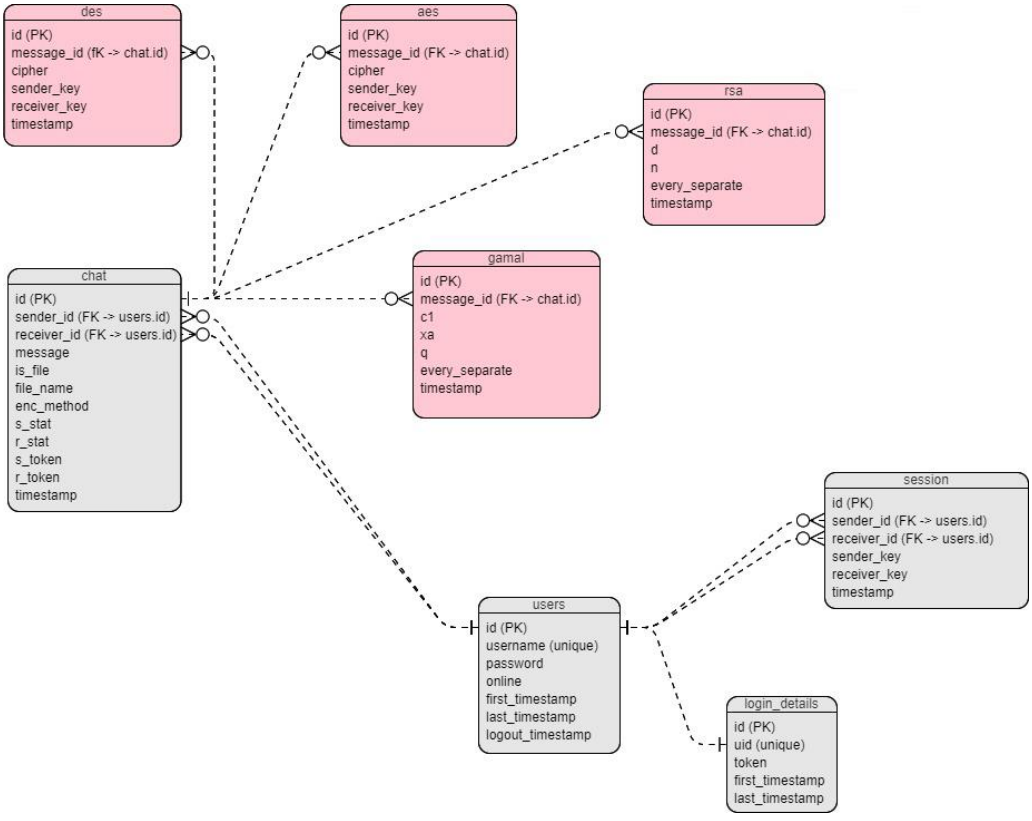


Рисунок 2.3 – Конструкція бази даних MySQL.

## 2.2. Реалізація взаємодії з сервером бази даних

В розділі 2, на рисунку 2.3 зображена потрібна логіка побудови бази даних MySQL. Для цього використаємо рідну мову програмування для реляційних баз даних – SQL[**6**].

Створимо таблицю «users»: містить дані про користувачів. Поля включають ‘id’(первинний ключ), ‘username’(унікальний), ‘password’, ‘online’, ‘first_timestamp’, ‘last_timestamp’, ‘logout_timestamp’.

Створимо таблицю «login_details»: зберігає інформацію про входи користувачів. Поля включають ‘id’(первинний ключ), ‘uid’(унікальний ідентифікатор користувача), ‘token’, ‘first_timestamp’, ‘last_timestamp’.

Створимо таблицю «session»: містить дані про сесії. Поля включають ‘id’(первинний ключ), ‘sender_id’, ‘receiver_id’, ‘sender_key’, ‘receiver_key’, ‘timestamp’.

Створимо таблицю «chat»: містить дані про чат-повідомлення. Наявні такі поля ‘id’(первинний ключ), ‘sender_id’, ‘receiver_id’, ‘message’, ‘is_file’, ‘filename’, ‘enc_method’, ‘s_stat’, ‘r_stat’, ‘s_token’, ‘r_token’, ‘timestamp’.

Далі створені бази для алгоритмів шифрування.

Таблиця «aes»: містить дані про повідомлення, що зашифровані алгоритмом AES. Поля включають ‘id’(первинний ключ), ‘message_id’, ‘cipher’, ‘sender_key’, ‘receiver_key’, ‘timestamp’.

Таблиця «des»: містить дані про повідомлення, що зашифровані алгоритмом DES. Поля включають ‘id’(первинний ключ), ‘message_id’, ‘cipher’, ‘sender_key’, ‘receiver_key’, ‘timestamp’.

Таблиця «gamal»: містить дані про повідомлення, що зашифровані алгоритмом El-Gamal. Поля включають ‘id’(первинний ключ), ‘message_id’, ‘c1’, ‘xa’, ‘q’, ‘every_separate’, ‘timestamp’.

Таблиця «rsa»: містить дані про повідомлення, що зашифровані алгоритмом RSA. Поля включають ‘id’(первинний ключ), ‘message_id’, ‘d’, ‘n’, ‘every_separate’, ‘timestamp’.

Нижче, на малюнку 2.4 наведено фрагмент коду бази даних, за допомогою якого було створено таблицю «chat».

```
CREATE TABLE `chat` (
  `id` int(11) NOT NULL,
  `sender_id` int(11) NOT NULL,
  `receiver_id` int(11) NOT NULL,
  `message` longtext NOT NULL,
  `is_file` tinyint(4) NOT NULL DEFAULT 0,
  `file_name` varchar(255) DEFAULT NULL,
  `enc_method` tinyint(4) NOT NULL DEFAULT 0,
  `s_stat` tinyint(4) NOT NULL DEFAULT 0,
  `r_stat` tinyint(4) NOT NULL DEFAULT 0,
  `s_token` varchar(255) NOT NULL,
  `r_token` varchar(255) NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Рисунок 2.4 – Фрагмент коду бази даних, створення таблиці «chat».

Для кожної таблиці було встановлено первинні ключі, також встановлено унікальні ключі для деяких полів, наприклад ‘username’ в таблиці «users».

Для кожної таблиці з первинним ключем типу ‘int’ був встановлений автоінкремент. Приклад для таблиці «session» на рисунку 2.5.

```
ALTER TABLE `session`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

Рисунок 2.5 – автоінкремент для таблиці «session».

## 2.3. Реалізація сервера чату

В даному розділі знаходиться розробка серверної частини чату.

У файлі chat_server.php написаний код, що потрібен для управління сеансом користувача в чаті. Він представляє функції правильного виходу з системи, генерації унікальних токенів для автентифікації. Використані стандартні методи роботи з сесіями та базами даних PHP[7]. Код на рисунку 2.6.

```php
<?php
  require_once('DB.php');
  $db = DB::getInstance();
  session_start();

  if(isset($_POST['logout']))
  {
      $uid = $_SESSION['uid'];
      $query = "UPDATE users SET online=0, logout_timestamp=CURRENT_TIMESTAMP() WHERE id=$uid";

      session_unset();
      session_destroy();

      if($db->query($query) === true){
        header("refresh:0;url=../");
      }
  }

  function generateRandomToken($db)
  {
      for(;;)
      {
          $length = 16;
          $word = array_merge(range('a', 'z'), range(0, 9), range('A', 'Z'));
          shuffle($word);
          $token = substr(implode($word), 0, $length);

          $query = "SELECT * FROM login_details WHERE token='$token'";
          $queryResult = mysqli_query($db, $query);
          if(mysqli_num_rows($queryResult) == 0)
              break;

      }
      return $token;
  }
?>
```

Рисунок 2.6 – Код chat_server.php.

Далі на PHP створимо код, що являтиме собою реалізацію класу ‘DB’ для роботи з базою даних. В роботі було використано патерн проектування Singleton. Код зображено на рисунку 2.7.

```php
<?php
    class DB
    {
        private static $instance;
        private $db_conn;

        private function __construct()
        {
            $this->db_conn = $this->databaseConnection();
            $this->databaseSelection();
        }

        private function databaseConnection()
        {
            if($db = mysqli_connect('localhost', 'root', '', 'sp_chat'))
            {
                return $db;
            }
            else
            {
                die(mysqli_error($db));
            }
        }

        private function databaseSelection()
        {
            mysqli_select_db($this->db_conn, 'sp_chat');
        }

        public static function getInstance()
        {
            if(!isset(self::$instance)){
                self::$instance = new DB();
            }
            return self::$instance->db_conn;
        }

    }
?>
```

Рисунок 2.7 – Код для класу ‘DB’.

Клас ‘DB’ реалізує патерн Singleton, що гарантує створення лиш одного з’єднання з базою даних протягом всього часу виконання скрипту. Це дозволяє суттєво економити ресурси та викорінює можливі проблеми з кількісними з’єднаннями з одною базою даних. Використання методів ‘databaseConnection’ та ‘databaseSelection’ спрощує управління підключеннями та вибором бази даних.

Написано код файлу server.php. Цей код реалізує серверну частину чата з функціями реєстрації, входу й виходу в систему. Фрагмент коду, що зображений на рисунку 2.8 показує, процес генерації токену. Генерується випадковий токен довжиною 16 символів, що складається з букв та цифр. Потім перевіряється унікальність токену, повторюючи процес, якщо згенерований токен вже знаходиться в базі даних. Потім просто повертає унікальний токен.

```php
function generateRandomToken($db)
{
    for(;;)
    {
        $length = 16;
        $word = array_merge(range('a', 'z'), range(0, 9), range('A', 'Z'));
        shuffle($word);
        $token = substr(implode($word), 0, $length);

        $query = "SELECT * FROM login_details WHERE token='$token'";
        $result = mysqli_query($db, $query);
        if(mysqli_num_rows($result) == 0)
            break;

    }
    return $token;
}
```

Рисунок 2.8 – Процес генерації випадкового токену.

Варто сказати, що паролі користувачів, що зберігаються в базі даних шифруються за допомогою алгоритму MD-5. Дивіться рисунок 2.9

```php
function specialEncryption($string){
  return md5($string);
}
```

Рисунок 2.9 – Шифрування за допомогою MD-5.

Тобто скрипт server.php реалізує основні функції для керування користувачами в чаті: реєстрацію, вхід, вихід, генерацію токенів. Він опрацьовує запити користувачів, взаємодіє з базою даних, керує сесіями та надає повідомлення про успішність реєстрації, або автентифікації на фронтенд.

## 2.4. Реалізація екрану логіну та реєстрації

Сторінка автентифікації «index.php» побудована з використанням HTNL, CSS та PHP. Вона включає в себе форму для вводу імені користувача та пароля з кнопкою відправки даних. Також є посилання на сторінку реєстрації для нових користувачів. Візуальна частина стилізована за допомогою Bootstrap та кастомних стилів, що забезпечує сучасний, чистий інтерфейс.

Підключений PHP скрипт для оброблення даних на сервері. JavaScript код запобігає повторній відправці форми при оновленні сторінки. Рисунок 2.10.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://kit.fontawesome.com/4733528720.js" crossorigin="anonymous"></script>

  <title>    red Chat</title>
  <link rel="stylesheet" type="text/css" href="resources/css/index.css">
  <link rel="stylesheet" type="text/css" href="resources/css/queries.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/normalize.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/Grid.css">
</head>
<body>
  <h1 class="d-flex justify-content-left">    red Chat</h1>
  <br/>
  <h1 style="text-align: center; font-size:64px;">
        Login
      </h1>

    <form action="" method="POST" style="margin: 20px auto;">
      <div class="form-group">
        <input type="text" class="form-control" placeholder="Username" required name="username">
      </div>

      <div class="input-group mb-3">
        <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password" name="password" required>
        <div class="input-group-append">
          <button type="submit" class="btn btn-outline-primary w-100" name="login">Login</button>
        </div>
      </div>
    </form>
    <div id="small" style="text-align: center;">
      <small>
        <a href="register.php" class="text-secondary">Register?</a>
      </small>
    </div>
    <?php require_once 'server/server.php'; ?>
</body>
</html>

<script>
    if ( window.history.replaceState ) {
      window.history.replaceState( null, null, window.location.href );
    }
</script>
```

Рисунок 2.10 – Код сторінки логіну «index.php».

Сторінка реєстрації «register.php» побудована за допомогою HTML, CSS, PHP. Вона включає в себе форму для вводу бажаного імені користувача та пароля, також є кнопка, що ініціює відправку даних на сервер. Наявне посилання для переходу на сторінку логіну. Візуальна частина сторінки стилізована за допомогою технології Bootstrap і кастомних стилів, що забезпечує сучасний та зручний інтерфейс оформлений в одному стилі зі сторінкою логіну.

Підключення PHP скрипта запускає обробку даних на сервері. JavaScript же в свою чергу запобігає повторній відправці форми при оновленні сторінки, покращуючи користувацький досвід. Код на рисунку 2.11.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://kit.fontawesome.com/4733528720.js" crossorigin="anonymous"></script>

  <title>    red Chat</title>
  <link rel="stylesheet" type="text/css" href="resources/css/index.css">
  <link rel="stylesheet" type="text/css" href="resources/css/queries.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/normalize.css">
  <link rel="stylesheet" type="text/css" href="vendors/css/Grid.css">
</head>
<body>
  <h1 class="d-flex justify-content-left">    red Chat</h1>
  <br/>
  <h1 style="text-align: center; font-size:64px;">
          Register
      </h1>

    <form action="" method="POST" style="margin: 20px auto;">
      <div class="form-group">
          <input type="text" class="form-control" placeholder="Username" required name="username">
      </div>

      <div class="input-group mb-3">
          <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password" name="password" required>
          <div class="input-group-append">
              <button type="submit" class="btn btn-secondary w-100" name="register">Register</button>
          </div>
      </div>
    </form>
    <div id="small" style="text-align: center;">
      <small>
          <a href="index.php" class="text-primary">Login?</a>
      </small>
    </div>
    <?php require_once 'server/server.php'; ?>
</body>
</html>

<script>
    if ( window.history.replaceState ) {
      window.history.replaceState( null, null, window.location.href );
    }
</script>
```

Рисунок 2.11 – Код сторінки реєстрації «register.php».

## 2.5. Реалізація екрану чату

Створено внутрішній файл «index.php», код якого представляє собою веб-сторінку для чату. В нього входять різні елементи, такі як модальне вікно для діалогу між користувачами, можливість відправки повідомлень та файлів, а також умовне відображення контенту в залежності від того, чи увійшов користувач в систему. В основі лежить комбінація технологій HTML, CSS та Javascript, яка забезпечує інтерактивний користувацький досвід.

Основні функції – це відображення різного контенту в залежності від статусу користувача. Відображення модального діалогового вікна для діалогу, включаючи надсилання повідомлень, файлів та вибір алгоритму шифрування. Забезпечення динамічної зміни розміру текстової області.

Фрагмент коду з реалізацією вибору елемента шифрування в модальному вікні, Нижньої частини модального вікна, та напису, що випадає в результаті спроби зайти на головну сторінку оминувши авторизацію – на малюнку 2.12.


Рисунок 2.12 – Фрагмент коду «index.php».

Написано код «check_receiver_status.php». Цей скрипт обробляє AJAX запити для отримання актуального статуса онлайн-користувачів, та часу їх останньої активності. За допомогою AJAX він взаємодіє з клієнтською стороною для оновлення інформації на веб-сторінці без повного її оновлення.

Приклад коду, для визначення часу відправки повідомлення на рисунку 2.13

```php
function time_elapsed_string($datetime, $full = false) {
    $now = new DateTime;
    $ago = new DateTime($datetime);
    $diff = $now->diff($ago);

    $diff->w = floor($diff->d / 7);
    $diff->d -= $diff->w * 7;

    $string = array(
        'y' => 'year',
        'm' => 'month',
        'w' => 'week',
        'd' => 'day',
        'h' => 'hour',
        'i' => 'minute',
        's' => 'second',
    );
    foreach ($string as $k => &$v) {
        if ($diff->$k) {
            $v = $diff->$k . ' ' . $v . ($diff->$k > 1 ? 's' : '');
        } else {
            unset($string[$k]);
        }
    }

    if (!$full) $string = array_slice($string, 0, 1);
    return $string ? implode(', ', $string) . ' ago' : 'just now';
```

Рисунок 2.13 – фрагмент коду «check_receiver_status.php»

Файл «display_messages» оброблює AJAX запити для отримання повідомлень чату між адресантом та адресатом. Скрипт виконує запити в базу даних, дістає повідомлення чата й оновлює їх статус, якщо це потрібно. Отримані дані передаються назад клієнтській стороні в форматі JSON.

Написано скрипт, що використовується для динамічного створення й оновлення списку користувачів на веб-сторінці та забезпечує можливість ініціювати чат з обраним користувачем. Код зображений нижче, на рисунку 2.14

```php
<?php
  require_once('../server/DB.php');
  $db = DB::getInstance();
  $uid = $_SESSION['uid'];

  $query = "SELECT * FROM users WHERE id != $uid ORDER BY username ASC";
  $result = mysqli_query($db, $query);
  $table = "<table class='table table-hover' id='users-table' style='margin: 40px auto; width:60%; box-shadow: -2px 6px 10px rgb(235, 235, 235); border-bottom: 1px solid #eee
            <thead class='table-dark'>
              <th>User</th>
              <th>Status</th>
              <th>Chat</th>
            </thead>";
  while($row = mysqli_fetch_assoc($result)){
    $rid = $row['id'];
    $username = $row['username'];
    $online = $row['online'];
    $last_timestamp = $row['last_timestamp'];

    $dot = "";
    $status = "Offline";

    if($online == 1){
      $dotColor = "#198853";
      $chatColor = "#0E6DFD";
      $status = "Online";
    }
    else{
      $dotColor = "#6C757D";
      $chatColor = "#6C757D";
      $status = "Offline";
    }

    $status = "<span class='receiver-public-statusText'>$status</span>";
    $dot = "<i style='font-size:10px; color:$dotColor; transition: all 3s;' class='fas fa-circle align-middle receiver-public-dot'></i>";

    $table .= "
    <tr>
      <td class='fw-light' style='padding-left: 20px;'>".ucfirst($username)."</td>
      <td>$status   $dot</td>
      <td>
        <button class='btn btn-sm border-0' data-bs-toggle='modal' data-bs-target='#chatModal' data-bs-senderId='$uid' data-bs-receiverId='$rid' data-bs-username='$username'
      </td>
      </tr>";
  }
  echo $table;
?>
```

Рисунок 2.14 – Код «get_users.php»

```php
<?php
    require_once('../server/DB.php');
    $db = DB::getInstance();

    $senderId = $_POST['sid'];
    $receiverId = $_POST['rid'];
    $senderKey = $_POST['skey'];
    $receiverKey = $_POST['rkey'];

    $query = "SELECT * FROM session
              WHERE (sender_id=$senderId OR sender_id=$receiverId) AND (receiver_id=$receiverId OR receiver_id=$senderId)";
    $result = mysqli_query($db, $query);
    $numOfRows = mysqli_num_rows($result);

    if($numOfRows > 0){
      $query = "UPDATE session
                SET sender_key='$senderKey', receiver_key='$receiverKey'
                WHERE (sender_id=$senderId OR sender_id=$receiverId) AND (receiver_id=$receiverId OR receiver_id=$senderId)";
      if($db->query($query) !== true){
        echo "Keys are not updated due to an error".mysqli_error($db);;
      }
    }
    else{
      $query = "INSERT INTO session (sender_id, receiver_id, sender_key, receiver_key)
      VALUES ('$senderId', '$receiverId', '$senderKey', '$receiverKey')";
      if($db->query($query) !== true){
        echo "Keys are not stored due to an error".mysqli_error($db);;
      }
    }
?>
```

Рисунок 2.15 – код «insert_both_keys.php»

На рисунку 2.15 зображено код файлу, що опрацьовує дані відправлені методом POST. Цей скрипт відповідає за забезпечення безпечного обміну ключами між відправником і отримувачем для шифрування й дешифрування повідомлень в захищеному чаті.

Написано «insert_message.php», він оброблює дані, відправлені методом POST, щоб зберегти повідомлення в базі даних з урахуванням обраного алгоритму шифрування.

Також написано два методи «typing_process.php» та «typing_ready.php» Вони забезпечують підключення до бази даних, що забезпечує доступ до даних, які скрипти можуть читати, виправляти, або видаляти по мірі необхідності. Код зображено на рисунку 2.16.

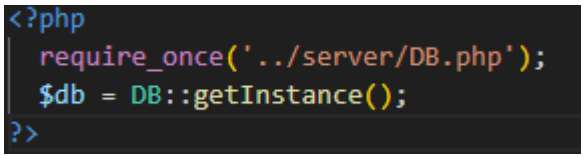


Рисунок 2.16 – метод для підключення БД.

# 3. РЕАЛІЗАЦІЯ ЗАХИСТУ ІНФОРМАЦІЇ ВЕБ-ЗАСОБУ ОБМІНУ ПОВІДОМЛЕННЯМИ

## 3.1. Реалізація алгоритму DES

Мета цього алгоритму – зашифрувати 64-бітні дані, використовуючи 56-бітний ключ. Він приймає на вхід 64 біти даних та ключа. DES[**8**] включає 16 раундів однакових операцій. Раунди DES включають розширення, операцію XOR з раундовим ключем, заміну та перестановку. Структура базується на мережі Фейстеля. Алгоритм включає в себе такі функції:

ConvertHexToBinary() – Конвертує з шістнадцяткових чисел у двійкові

ConvertBinaryToHex() – Конвертує з двійкових чисел у шістнадцяткові

ConvertBinaryToDecimal() – Конвертує з двійкових чисел у десяткові

ConvertDecimalToBinary() – Конвертує з десяткових чисел у двійкові

Permutation() – Функція перестановки для реорганізацій бітів

ShiftLeft() – Зсув бітів вліво на n позицій відповідно до раундів

XOR() – Обчислення операції XOR для двох рядків двійкових чисел

Encryption(): -Конвертувати з шістнадцяткових чисел у двійкові.

-Використати функцію перестановки.

-Розділити 64 біти на два масиви лівий та правий.

-Розширити 32 біти даних до 48 бітів.

-Заміна значення з таблиці S обчисленням рядка та стовпця.

-Після заміни використати функцію перестановки.

-Виконати XOR для лівого і sbox_string.

-Поміняти ліві та праві місцями.

-Об’єднати ліві і праві.

-Реорганізація бітів для отримання шифрованого тексту.

Decryption() – Виконати ті ж кроки, що й для шифрування, використовуючи підключі в зворотному порядку SK16>SK1. Перший раунд з SK16 скасовує 16й раунд шифрування та відновлює оригінальний відкритий текст.

В цій роботі не було використано бібліотеки. Всі алгоритми шифрування були написані з нуля. Код php для алгоритму DES на рисунку 3.1

```php
class DES{
    function text2bin($text)
    {
        $binstr = "";
        for($i =0; $i<strlen($text) ; $i++)
        {
            $temp =base_convert($text[$i],16,2);
            $temp= str_pad($temp,4,'0',STR_PAD_LEFT);

            $binstr .= $temp;
        }

        return $binstr;
    }
    function premute_PC1($a)
    {
        $table = array(57, 49, 41, 33, 25, 17, 9,
                        1, 58, 50, 42, 34, 26, 18,
                       10,  2, 59, 51, 43, 35, 27,
                       19, 11,  3, 60, 52, 44, 36,
                       63, 55, 47, 39, 31, 23, 15,
                        7, 62, 54, 46, 38, 30, 22,
                       14,  6, 61, 53, 45, 37, 29,
                       21, 13,  5, 28, 20, 12,  4);

        $newstr ="";
        for($i=0 ; $i<56; $i++)
        {
            $index = $table[$i]-1;

            $newstr .= $a[$index];
        }
        return $newstr;
    }

    function Left_Shifting($C,$D)
    {
        $all = array();
```

Рисунок 3.1 – Фрагмент коду алгоритму DES.

## 3.2. Реалізація алгоритму AES

Специфікація Advanced Encryption Standard (AES)[**9**] була розроблена Національним інститутом стандартів і технологій США у 2001 році. AES широко використовується сьогодні, оскільки є набагато сильнішим, ніж DES і Triple DES, незважаючи на те, що його важче реалізувати. Раунди нижче на рисунку 3.2

Використовуваний розмір секретного ключа становить 128 бітів (16 байт), тобто для шифрування/дешифрування відкритого тексту потрібно 10 раундів.

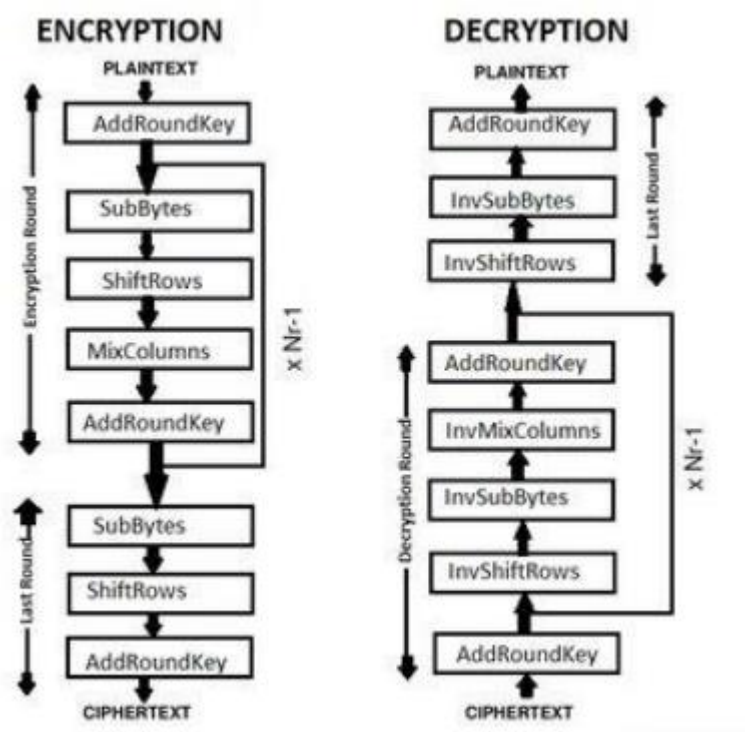


Рисунок 3.2 – Графічна схема алгоритму AES.

Розширення ключа AES:

Спочатку ми розбиваємо ключ на чотири блоки, потім на четвертому блоці w[3] виконуємо: -циклічний зсув байтів вліво

-заміну байтів S-box

-XOR з таблицею констант раунду = g(w[3])

Потім виконуємо операцію XOR,щоб згенерувати 4 нові блоки з отриманого блоку g(w[4]) для формування нового ключа:

w[4] = w[0] XOR g(w[3])

w[5] = w[1] XOR w[4]

w[6] = w[2] XOR w[5]

w[7] = w[3] XOR w[6]

Об’єднуючи всі блоки від w[4] до w[7] формуємо раундовий ключ. У шифруванні ми генеруємо нові раундові ключі паралельно з кожною ітерацією матриці стану, щоб виконати XOR з кожним згенерованим раундовим ключем. У дешифруванні ми спочатку генеруємо всі раундові ключі, потім виконуємо XOR матриці стану з раундовими ключами у зворотному порядку в кожній ітерації. Всього 10 ітерацій, адже використовуємо 128 біт.

Додавання раундового ключа до матриці стану:

Матриця стану – це відкритий текст у вигляді двомірного масиву.

Ми просто використовуємо операцію XOR з поточним раундовим ключем, щоб створити нову матрицю стану в кожному раунді, доки на 10му раунді не буде отримано шифр.

S-Box та зворотний S-Box:

Ми просто замінюємо байти використовуючи S-Box Rijndael на рисунку 3.3

AES S-box

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Inverse S-box

| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0a | 0b | 0c | 0d | 0e | 0f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 52 | 09 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| 10 | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| 20 | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| 30 | 08 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| 40 | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| 50 | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| 60 | 90 | d8 | ab | 00 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 05 | b8 | b3 | 45 | 06 |
| 70 | d0 | 2c | 1e | 8f | ca | 3f | 0f | 02 | c1 | af | bd | 03 | 01 | 13 | 8a | 6b |
| 80 | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| 90 | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| a0 | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1b |
| b0 | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| c0 | 1f | dd | a8 | 33 | 88 | 07 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| d0 | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |
| e0 | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| f0 | 17 | 2b | 04 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

Рисунок 3.3 – Таблиця матриць.

ShiftRow та зворотній ShiftRow:

Перший рядок не зміщується.

Другий рядок зміщується на один раз вліво.

Третій рядок зміщується на два рази вліво.

Четвертий рядок зміщується на три рази вліво.

Mix Columns та зворотній Mix Columns:

Кожен рядок у матриці стану обробляється як чотиричленний поліном. Таблиця 1

Таблиця 1.

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Множення на матрицю.

У зворотному Mix Columnsми множимо на іншу матрицю..

Після шифрування/дешифрування:

Таблиця 2

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Вихідний текст буде конвертовано в рядок шістнадцяткових чисел, а потім назад у двійковий текст.

Весь алгоритм показано на рисунках 3.4, 3.5

```
class AES{

    function keyGen($key, int $rounds)
    {
        $nibble =     array(0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f);
        $sboxKey  =   array(0x09,0x04,0x0a,0x0b,0x0d,0x01,0x08,0x05,0x06,0x02,0x00,0x03,0x0c,0x0e,0x0f,0x07);


        $keys = array();

        $allkeys = array();

        $w = str_split($key,8);
        $ct = 0;
        $w0 = $w[0];
        $w1 = $w[1];
```

Рисунок 3.4 – Фрагмент коду алгоритму AES.

```
$mul3 = array( 0x00,0x03,0x06,0x05,0x0c,0x0f,0x0a,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
               0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
               0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x74,0x77,0x72,0x71,
               0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
               0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
               0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
               0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
               0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
               0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
               0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
               0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
               0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
               0x5b,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
               0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
               0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
               0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a);




for($i = 0 ; $i<count($M) ; $i++)
{
    for($j = 0 ; $j<count($M[$i]) ; $j++)
    {
        $M[$i][$j] = dechex($M[$i][$j]);
    }
}

for($i = 0 ; $i<count($mul2) ; $i++)
{
    $mul2[$i] = dechex($mul2[$i]);
}

for($i = 0 ; $i<count($mul3) ; $i++)
{
    $mul3[$i] = dechex($mul3[$i]);
}


$row = count($xa);
$col = count($xa[0]);

$result = array_fill(0, $row, array_fill(0, $col, 0));
```

Рисунок 3.5 – Фрагмент коду алгоритму AES

### 3.3. Реалізація алгоритму RSA

RSA[**10**] – це криптосистема з відкритим ключем, яка широко використовується для безпечної передачі даних. Це також одна з найстаріших криптосистем. Акронім походить від прізвищ Рона Рівеста, Аді Шаміра і Леонарда Адлемана, які опублікували алгоритм у 1977 році. Завдяки проблемі факторизації незламний.

-Спочатку p і q генеруються як прості числа за допомогою функцій ‘findRandomPrime()’ і ‘isPrime()’, де p! = q.

-N обчислюється шляхом множення p на q функцією ‘compute_n()’.

-ϕ(n), або z, обчислюється шляхом множення (p-1) на (q-1), за допомогою функції ‘eular_z()’.

-Потім генерується e за допомогою функції ‘find_e(0)’, яка перевіряє число e>1i<ϕ(n), і при цьому gcd(e, ϕ(n)) повинно дорівнювати 1.

-d генерується за допомогою функції ‘find_d()’, яка працює в нескінченому циклі, поки не знайде число d таке, що (d*e) mod ϕ(n) =1.

-M, яке представляє повідомлення, досягається шляхом перетворення символів ASCII-числа за допомогою функції ‘ord()’, при цьому M<n

-C, тобто шифротекст, генерується за допомогою функції ‘encrypt()’. У цій функції C конкатенується з функцією ‘bcpowmod()’ PHP, яка використовує метод піднесення до степеня та множення для піднесення e до отриманого ASCII-числа символу з M, потім mod n, повторюючи це для кожного символу, і конкатенує C.

-Змінна під назвою ‘everySeparate’ конкатенується з результатом для кожного C після кожного обчислення ‘bcpowmod()’, щоб допомогти при дешифруванні через різну довжину ASCII-цифр, наприклад, символів.

-Дешифруванні C, d, n, ‘everySeparate’ витягуються з бази даних, генеруючи M за допомогою‘chr(bcpowmod(current(c), d, n))’ і конкатенуючи кожну літеру до M (функція ‘chr()’ використовуються для перетворення ASCII-числа у символ

Весь алгоритм було написано на та показано нижче на рисунку 3.6, 3.7

```php
class RSA{

  function __construct() {}

  function findRandomPrime($p)
  {
      $min = 2;
      $max = 999;
      for ($i=rand($min, $max); $i < $max; $i++) {
          if ($this->isPrime($i) && $i != $p) {
              return $i;
          }
      }
  }

  function isPrime($num){
      if($num % 2 == 0) {
          return false;
      }

      for($i = 3; $i <= ceil(sqrt($num)); $i = $i + 2) {
          if($num % $i == 0)
              return false;
      }
      return true;
  }

  function compute_n($p, $q){
    return $p * $q;
  }

  function eular_z($p, $q){
    return ($p - 1) * ($q - 1);
  }

  function find_e($z){
    for($i = 2; $i < $z; $i++){
      if($this->coprime($i, $z)){
        return $i;
      }
    }
  }

  function gcd($e, $z){
      if ($e == 0 || $z == 0)
```

Рисунок 3.6 – Фрагмент коду алгоритму RSA.

```
class RSA{
constructor() {}

  findRandomPrime(p){
      const min = 300;
      const max = 999;
      for (;;) {
        const i = Math.floor(Math.random() * max) + min;
        if (this.isPrime(i) && i != p) {
            return i;
        }
      }
  }

  isPrime(num){
    if(num % 2 == 0) {
        return false;
    }

    for(let i = 3; i <= Math.ceil(Math.sqrt(num)); i = i + 2) {
        if(num % i == 0)
            return false;
    }
    return true;
  }

  compute_n(p, q){
    return p * q;
  }

  eular_z(p, q){
    return (p - 1) * (q - 1);
  }

  find_e(z){
    for(let i = 2; i < z; i++){
      if(this.coprime(i, z)){
        return i;
      }
    }
  }

  gcd(e, z){
      if (e == 0 || z == 0){
          return 0;
      }

      if (e == z){
```

Рисунок 3.7 – JS код алгоритму.

### 3.4. Реалізація алгоритму Diffie-Hellman

Алгоритм Diffie-Hellman[**11**] призначений для генерації зашифрованих секретних ключів, які поділяються тільки між двома користувачами. Ці ключі потім використовуються іншими алгоритмами шифрування тексту.

-findRandomPrime() – Використовується для генерації випадкового простого числа за допомогою функції ‘isPrime()’ (q).

-findPrimitives() – Використовується для генерції первісного кореня простого числа та вибору одного випадкового з них (a).

-Math.random() – Використовується для випадкової генерації приватних ключів < q (Xa, Xb).

-mpmod() – Використовується для обчислення публічних ключів (Ya, Yb) шляхом піднесення a до степеня xa, xb за модулем q методами квадратів.

-mpmod() – Використовується для обчислення секретних ключів (_a,_b) на основі методу квадратів і множення.

На рисунках 3.8 та 3.9 зображено код цього алгоритму.

```
class DiffieHellman{
  constructor() { }

  findRandomPrime(min, max){
    for (;;) {
      const i = Math.floor(Math.random() * max) + min;
      if (this.isPrime(i)) {
          return i;
      }
    }
  }

  isPrime(num){
    if(num % 2 == 0) {
        return false;
    }

    for(let i = 3; i <= Math.ceil(Math.sqrt(num)); i = i + 2) {
        if(num % i == 0)
            return false;
    }
    return true;
  }
```

Рисунок 3.8 – Фрагмент коду алгоритму.

```
mpmod(base, exponent, modulus) {
  if ((base < 1) || (exponent < 0) || (modulus < 1)) {
      return ("invalid");
  }
  let result = 1;
  while (exponent > 0) {
      if ((exponent % 2) == 1) {
          result = (result * base) % modulus;
      }
      base = (base * base) % modulus;
      exponent = Math.floor(exponent / 2);
  }
  return (result);
}

findPrimitives(theNum) {
  var o = 1;
  var k;
  var roots = new Array();
  var z = 0;

  for (var r = 2; r < theNum; r++) {
    k = Math.pow(r, o);
    k %= theNum;
    while (k > 1) {
      o++;
      k *= r;
      k %= theNum;
    }
    if (o == (theNum - 1)) {
      roots[z] = r;
      z++;
    }
    o = 1;
  }

  return roots;
}
```

Рисунок 3.9 – JS  код методу шифрування.

Код цього алгоритму повністю виконано на Java Script.

## 3.5. Реалізація алгоритму El-Gamal

Система шифрування El-Gamal[**12**] є асиметричною криптосистемою з відкритим ключем, заснованою на обміні ключами Diffie-Hellman. Її описав Тагер Ель Гамал у 1985 році. Ця криптосистема використовує асиметричне шифрування для обміну повідомленнями між двома сторонами. Основу становить складність знаходження дискретного логарифму в циклічній групі.

Процес генерації ключів:

Генерація простого числа q і його первісного кореня a:

-Генеруємо велике випадкове число q.

-Знаходимо випадкове число a, яке є первісним коренем q і меншим за q.

Вибір приватного ключа Xa та генерація публічного ключа Ya:

-Вибираємо приватний ключ Xa випадковим чином, він менший від q.

-Генеруємо публічний ключ Ya.

Функція шифрування.

Функція дешифрування

Додаткові деталі:

Використовується метод ”Every Separate” для обробки більше ніж одного символу в повідомленні (див пункт 4.3 RSA).

Літери перетворюються у їх ASCII-десяткові коди, виконується алгоритм, і в кінці результат перетворюється назад у символи.

El-Gamal досі вважається незламним завдяки проблемі дискретного логарифму.

Нижче на рисунках 3.10, 3.11 та 3.12 показано код на Java Script.

```
class Gamal{

  __construct() {}

  findRandomPrime(){
    const min = 299;
    const max = 999;
    for (;;) {
      const i = Math.floor(Math.random() * max) + min;
      if (this.isPrime(i)) {
          return i;
      }
    }
  }
```

Рисунок 3.10 – Фрагмент коду алгоритму.

```
encrypt(q, a, ya, message){
  var ciphers = new Object();
  let everySeparate = "";

  let k1 = Math.floor(Math.random() * q - 1) + 2;
  let k2 = this.mpmod(ya, k1, q);
  let c1 = this.mpmod(a, k1, q);
  let c2 = "";

  for(let i = 0; i < message.length; i++){
    let currentChar = message.charCodeAt(i);
    let currentc2 = (k2 * parseInt(currentChar)) % q;
    c2 += currentc2;
    everySeparate += String(currentc2).length;
  }

  ciphers[0] = c1;
  ciphers[1] = c2;
  ciphers[2] = everySeparate;
  return ciphers;
}

decrypt(c1, c2, xa, q, everySeparate){
  let m = "";
  let k2 = this.mpmod(c1, xa, q);
  let k2Inverse = this.modInverse(k2, q);

  for(let i = 0, ct = 0; i < String(c2).length; i+=parseInt(everySeparate[ct]), ct++){
    m += this.getTheCurrentChar(String(c2), i, parseInt(everySeparate[ct]), k2Inverse, q);
  }
  return m;
}
```

Рисунок 3.11 – Фрагмент коду.

```
isPrime(num){
  if(num % 2 == 0) {
      return false;
  }

  for(let i = 3; i <= Math.ceil(Math.sqrt(num)); i = i + 2) {
      if(num % i == 0)
          return false;
  }
  return true;
}
```

Рисунок 3.12 – Фрагмент коду.

# 4. ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ВЕБ-ЗАТОСУНКУ

В цьому розділі проведено тестування готового й скомпонованого осередку.

Першим пунктом проведемо перевірку спроби зайти в неіснуючий обліковий запис. Введемо неіснуючий логін ‘Zhenia’ та пароль ‘password’.

В результаті отримуємо напис ‘Wrong Username or Password’ –рисунок 4.1.

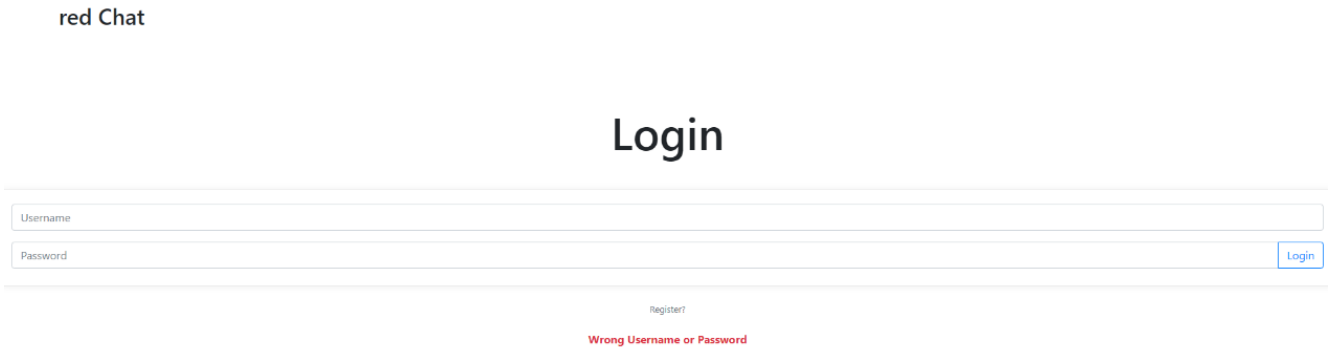


Рисунок 4.1 – Тест неіснуючого користувача.

Тепер перейдемо на сторінку реєстрації й зареєструємо новий акаунт з нікнеймом ‘Zhenia’ та паролем ’password’. Потім знову зайдемо на вкладку логіна й введемо логін та пароль, проте пароль введемо починаючи з великої букви. В результаті розуміємо, що пароль чутливий до регістру, як і нікнейм. Дивіться рисунок 4.2

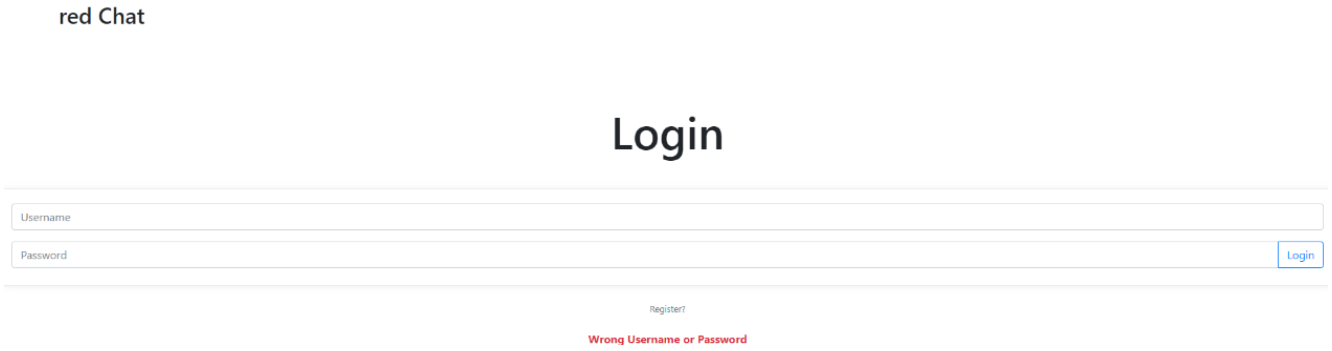


Рисунок 4.2 – Перевірка чутливості до регістру.

Тепер введемо вірний логін та пароль, в результаті нас повинно пустити на головну сторінку. Рисунок 4.3.

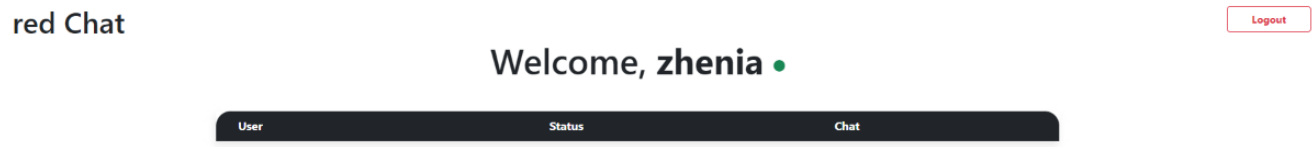


Рисунок 4.3 – Вхід до головного вікна.

Як бачимо, в нас немає активних контактів. Це тому що більше немає зареєстрованих користувачів. Виправимо це. Зареєструємо користувача з ніком ‘Dzhaelit Rakata’ та паролем ‘1234567890’. В результаті, коли увійдемо в акаунт в списку доступних контактів з’явиться Джаеліт. Дивіться на рисунку 4.4.

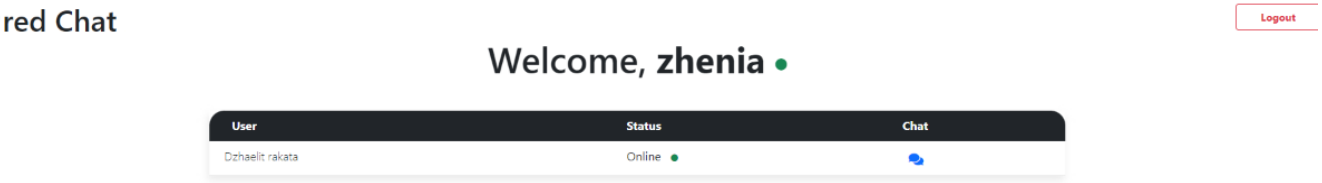


Рисунок 4.4 – Тест працездатності списку користувачів.

Як бачимо індикатор зеленою крапкою сигналізує, що користувач онлайн.

Також в адміністративній панелі бази даних побачимо активовані акаунти користувачів. Буде видно їх ніки, проте паролі будуть зашифровані. Ще там є інформація про дату реєстрації профілю, про час останньої активності Рисунок 4.5.

| id | username | password | online | first_timestamp | last_timestamp | logout_timestamp |
|---|---|---|---|---|---|---|
| 18 | dzhaelit rakata | e807f1fcf82d132f9bb018ca6738a19f | 1 | 2024-05-19 02:16:25 | 2024-05-19 02:16:33 | 2024-05-19 02:16:25 |
| 19 | zhenia | 5f4dcc3b5aa765d61d8327deb882cf99 | 1 | 2024-05-19 02:16:48 | 2024-05-19 02:16:56 | 2024-05-19 02:16:48 |

Рисунок 4.5 – База даних MySQL

Клікнувши на синю іконку чату перейдемо до вікна листування, Рисунок 4.6.

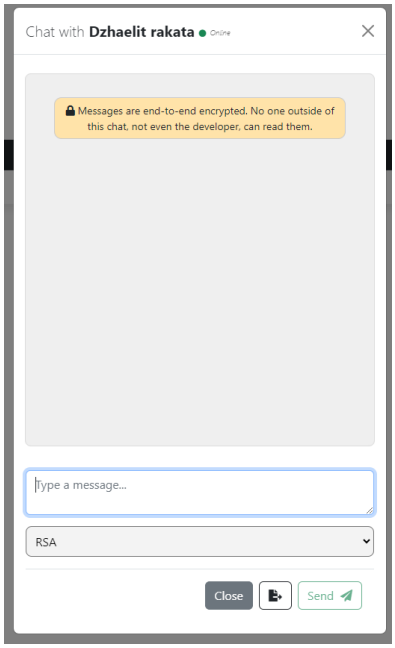


Рисунок 4.6 – Тест безшовного переходу у вікно чату.

Як і в дизайні присутні всі потрібні елементи. Клікнемо на вкладку вибору типу шифрування, в базі там завжди обрано RSA. Рисунок 4.7.

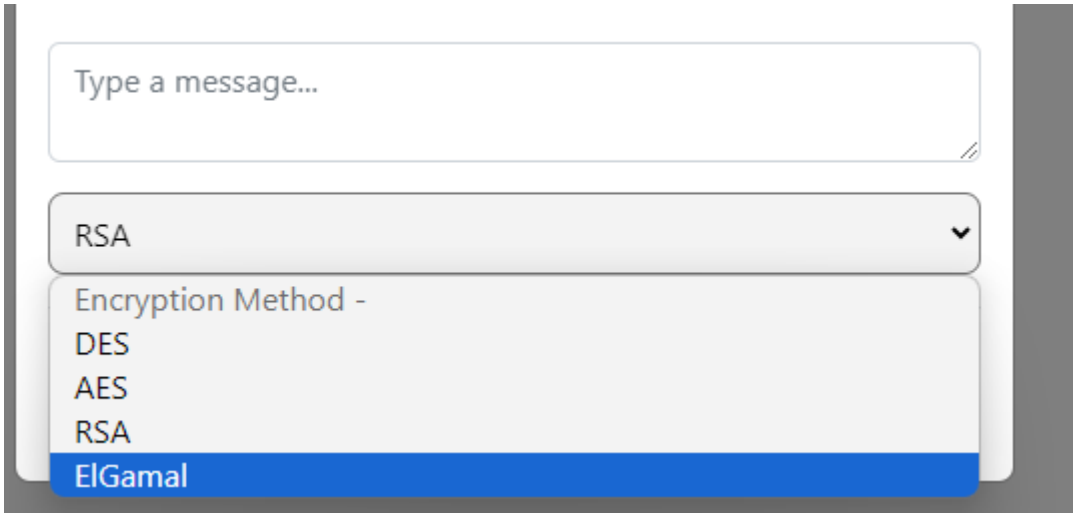


Рисунок 4.7 – Працездатність списку вибору алгоритму.

Оберемо для прикладу алгоритм шифрування El-Gamal.

Тепер з акаунта Джаеліт відправимо текстове повідомлення. Рисунок 4.8.

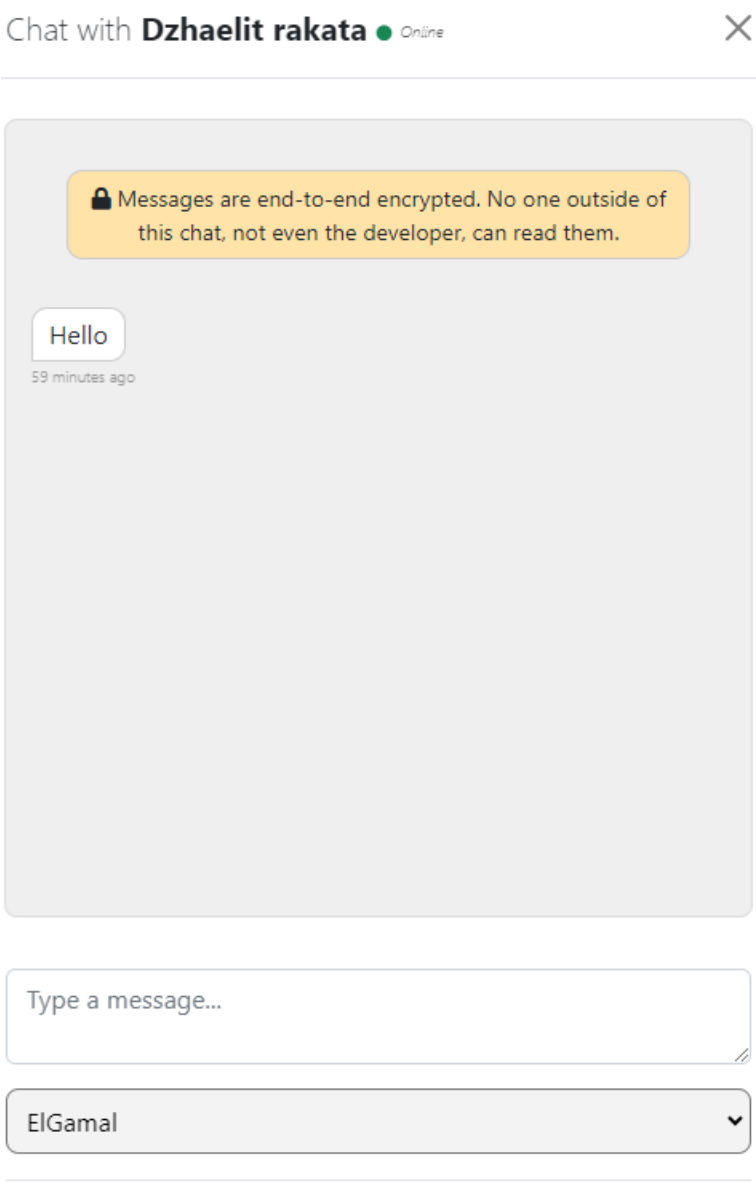


Рисунок 4.8 – Перевірка відправки й відображення повідомлень.

Бачимо Коректне відображення повідомлення в отримувача.

Напишемо відповідь Джаеліт і в адміністраторській панелі бази даних побачимо інформацію про повідомлення. Рисунок 4.9.

| id | message_id | c1 | xa | q | every_separate | timestamp |
|---|---|---|---|---|---|---|
| 50 | 392 | 38 | 618 | 1231 | 33334 | 2024-05-19 02:31:07 |
| 51 | 393 | 402 | 812 | 857 | 333333233 | 2024-05-19 02:33:26 |

Рисунок 4.9 – Шифровані дані в базі.

Тут є інформація про точний час відправлення повідомлення, проте саме повідомлення передається числовими значеннями. Ключ до яких знайти неможливо, адже він додатково шифрується ы передається напряму від користувача користувачу за допомогою алгоритму Diffie-Hellman.

Тепер спробуємо надіслати інформацію файлом, див. рисунок 4.10.

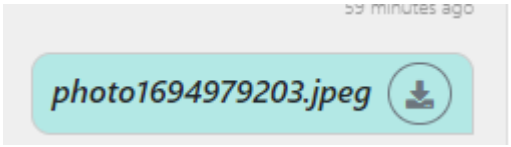


Рисунок 4.10 – Перевірка відправки й отримання файлу.

В адміністраторській панелі знову бачимо шифроване повідомлення, причому картинка дійшла до кінцевого користувача абсолютно неушкодженою, що демонструє абсолютно точність роботи алгоритму шифрування/дешифрування.

Провівши тести було виявлено, що й інші алгоритми, а саме DES, AES, RSA працюють відповідно до вимог. Ні в одному з випадків неможливо будь-яким чином розшифрувати суть повідомлення, навіть маючи прямий доступ до бази даних.

# ВИСНОВКИ

Розроблена система обміну повідомленнями з наскрізним шифруванням, що мінімізує збір та виток метаданих. Запропонована система інтегрує чотири криптографічні алгоритми, дозволяючи користувачам обирати метод залежно від власних побажань, специфіки використання та ресурсних обмежень.

Наскрізне шифрування, де всі повідомлення шифруються на стороні відправника, та розшифровуються стороною отримувача, що повністю виключає можливість їх перегляду третьою стороною, включаючи постачальника послуг.

Мінімізація збору метаданих, де зберігаються лише критично необхідна інформація для функціонування сервісу без зберігання вмісту.

Впроваджено додаткову систему шифрування ключів за допомогою використання стороннього алгоритму.

Таким чином запропонована архітектура здатна значно підвищити рівень конфіденційності та безпеки особистої інформації в сучасному цифровому середовищі.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Dewes, Christian, Arne Wichmann, and Anja Feldmann. "An analysis of Internet chat systems." Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement. 2003.
2. Beaird, Jason, Alex Walker, and James George. The principles of beautiful web design. SitePoint Pty Ltd, 2020.
3. https://explodingtopics.com/blog/messaging-apps-stats
4. Fuentes, Mayra Rosario. "Cybercrime and other threats faced by the healthcare industry." Trend Micro 5566 (2017).
5. Josefsson, Simon. The base16, base32, and base64 data encodings. No. rfc4648. 2006.
6. Welling, Luke, and Laura Thomson. PHP and MySQL Web development. Sams publishing, 2003.
7. Gope, Dibakar, David J. Schlais, and Mikko H. Lipasti. "Architectural support for server-side PHP processing." 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2017.
8. Grabbe, J. Orlin. "The DES algorithm illustrated." (2010).
9. Abdullah, Ako Muhamad. "Advanced encryption standard (AES) algorithm to encrypt and decrypt data." Cryptography and Network Security 16.1 (2017):
10. Zhou, Xin, and Xiaofei Tang. "Research and implementation of RSA algorithm for encryption and decryption." Proceedings of 2011 6th international forum on strategic technology. Vol. 2. IEEE, 2011.
11. Carts, David A. "A review of the Diffie-Hellman algorithm and its use in secure internet protocols." SANS institute 751 (2001): 1-7.
12. Imran, Omar A., et al. "Implementation of El-Gamal algorithm for speech signals encryption and decryption." Procedia Computer Science 167 (2020): 1028-1037.