Міністерство освіти і науки України

Харківський національний університет імені В.Н.Каразіна

Факультет комп’ютерних наук

Спеціальність 125 «Кібербезпека»

Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»
Зав.кафедрою БІСТ
Ольга МЕЛКОЗЬОРОВА
_____
« » 2023 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Дослідження методу підвищення захищеності месенджеру Telegram на основі стеганографічних алгоритмів»

оцінка « »

Голова ЕК

Олександр ЛЕМЕШКО

_____

Керівник: к.т.н., с.н.с. Сватовський І.І.

Рецензент: к.т.н., доц. Бакуменко Н.С.

Виконавець: студент(ка) групи КБ-61

Алєксєєв Артем Юрійович

Харків – 2023

РЕФЕРАТ

Пояснювальна записка містить 64 сторінок, 17 рисунків, 11 таблиць, 1 додаток, 57 джерел.

Метою дипломної роботи є дослідження методів підвищення захищеності месенджера Telegram за допомогою стеганографічних алгоритмів.

Об’єктом дослідження дипломної роботи є методи та алгоритми передачі повідомлень в месенджері Telegram.

Предметом розробки є Telegram бот, що реалізує метод підвищення захищеності месенджера Telegram за допомогою стеганографічних алгоритмів.

Методи дослідження: пошук та аналіз інформації стосовно предмету дослідження по у відкритих джерелах, узагальнення знайденої інформації, моделювання, розробка та практичне тестування стеганографічного алгоритму передачі повідомлень в мессенджері Telegram.

Результатами проведеної роботи є: аналіз захищеності месенджеру Telegram, визначені його можливі вразливості розробка Telegram бота, що реалізує метод підвищення захищеності месенджера Telegram на основі застосування стеганографічних алгоритмів.

Ключові слова: TELEGRAM, ЗАХИСТ ІНФОРМАЦІЇ, ЧАТ-БОТ, СТЕГАНОГРАФІЯ, WAV, PYTHON, МОТОД ФАЗОВОГО КОДУВАННЯ, АУДІОСИГНАЛ.

ABSTRACT

The explanatory note contains 64 pages, 17 figures, 11 tables, 1 appendix, and 57 sources.

The purpose of the thesis is to study methods of increasing the security of the Telegram messenger using steganographic algorithms.

The object of research of the thesis is methods and algorithms for transmitting messages in the Telegram messenger.

The subject of development is a Telegram bot that implements a method of increasing the security of the Telegram messenger using steganographic algorithms.

Research methods: search and analysis of information on the subject of research in open sources, generalization of the information found, modeling, development and practical testing of a steganographic algorithm for messaging in the Telegram messenger.

The results of the work are: analysis of the security of the Telegram messenger, identification of its possible vulnerabilities development of a Telegram bot that implements a method for increasing the security of the Telegram messenger based on the use of steganographic algorithms.

Keywords: TELEGRAM, INFORMATION SECURITY, CHATBOT, STEGANOGRAPHY, WAV, PYTHON, PHASE CODING MODAL, AUDIO SIGNAL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА СИМВОЛІВ 6
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Аналіз основних моделей зловмисників 10
1.2 Аналіз захищеності Telegram 11
1.3 Аналіз можливих підходів підвищення захищеності Telegram 17
1.4 Аналіз найбільш придатних типів стегано контейнерів 20
1.4.1 Порівняльний аналіз основних методів аудіо стеганографії 23
1.4.2 Аналіз стеганографічного методу фазового кодування 26
1.4.3 Аналіз форматів аудіо стегоконтейнерів 27
1.5 Чат-бот як механізм підвищення захищеності Telegram 29
2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 30
2.1 Розробка концептуальної моделі програмного забезпечення 30
2.2 Проектування патерну системної архітектури 34
2.3 Розробка специфікації системних вимог 37
2.4 Розробка архітектури програмного забезпечення 38
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 40
3.1 Вимоги до Telegram-бота 40
3.2 Обрання технологій розробки 42
3.2.1 Аналіз платформи для розробки 42
3.2.2 Аналіз фреймворків 45
3.2.3 Вибір хостингу 47
3.2.4 Середа розробки 49
3.3 Серверна частина програмного забезпечення 50
3.3.1 Реєстрація Telegram боту 50
3.3.2 Розробка Python проекту 51
3.3.3 Реалізація клієнтської частини боту 52
3.3.4 Реалізація серверної частини боту 55

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО TELEGRAM-БОТА 61
4.1 Розробка методики експерименту 61
4.1.1 Реалізація PSNR тестування 62
4.1.2 Реалізація BER тестування 67
4.2 Оцінка набору тестових даних 68
4.3 Аналіз результатів експерименту 68
ВИСНОВКИ 70
ПЕРЕЛІК ВИКОРСТАНИХ ДЖЕРЕЛ 72
ДОДАТОК А 80

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА СИМВОЛІВ

ACM	-	Association for Computing Machinery
E2EE	-	End-to-End Encryption
API	-	Application Programming Interface
HTTPS	-	HyperText Transfer Protocol Secure
TLS	-	Transport Layer Security
HTTP	-	HyperText Transfer Protocol
TCP	-	Transmission Control Protocol
UDP	-	User Datagram Protocol
CVE	-	CommonVulnerabilities and Exposures
2FA	-	Two-Factor Authentication
ЦОС	-	Цифрова обробка сигналів
ССЛ	-	Слухова система людини
ЗСЛ	-	Зорова система людини
LSB	-	Least Significant Bit
PSNR	-	Peak signal-to-noise ratio
ДПФ	-	Дискретне перетворення Фур'є
WAV	-	Waveform audio format
AAC	-	Advanced Audio Coding
MP3	-	Формат файлу для зберігання аудіоінформації
FLAC	-	Free Lossless Audio Codec
AIFF	-	Audio Interchange File Format
ПЗ	-	Програмне Забезпечення
RUP	-	Rational Unified Process
UML	-	Unified Modeling Language
SDLC	-	Synchronous Data-Link Control
ПС	-	Програмна система

БД	-	База даних
CBA	-	Component Based Architecture
JSON	-	JavaScript Object Notation
JS	-	JavaScript
CSS	-	Cascading Style Sheets
FSM	-	Finite State Machine
VDS	-	Virtual dedicated server
VPS	-	Virtual private server
ЄС	-	Європейський Союз
IDE	-	Integrated Development Environment
ШІ	-	Штучний інтелект
SSH	-	Secure SHell
BER	-	Bit Error Ratio
MSE	-	Mean Squared Error

ВСТУП

На сьогоднішній день, мережа Інтернет заполоняє все більше простору в житті людини. Найважливіші її сфери стали дуже тісно пов'язані з повсякденним існуванням. Не винятком є і сфера спілкування. Месенджери та соціальні мережі є частиною сьогоднішньої культури. Люди транслюють своє життя в соціальних мережах, обмінюються повідомленнями, в приватних та групових чатах. І якщо в соціальних мережах користувачі зазвичай обмінюються розважальною інформацією, самостійно показуючи моменти свого існування, месенджери часто використовують для роботи, навчання, ділових переписок, або навіть як програми для зберігання та обміну файлами, або даними що представляють конфіденційну інформацію. Такі дані користувачів стають мішенню для різних зловмисників. В цьому контексті стає важливим питання забезпечення їх захисту та при необхідностті практичній реалізації для цього програмних рішень.

Постановка задачі кваліфікаційної роботи

Об'єктом дослідження є месенджер Telegram та застосування стеганографічних алгоритмів для підвищення його захищеності.

Предмет дослідження - стеганографічні алгоритми та їх застосування для підвищення безпеки месенджера Telegram.

Метою дослідження є підвищення захищеності месенджеру Telegram за допомогою стеганографічних алгоритмів.

Метою дипломної роботи є дослідження методів підвищення захищеності месенджера Telegram за допомогою стеганографічних алгоритмів.

Для досягнення мети дослідження необхідно виконати перелічені задачі дослідження:

- проаналізувати захищеність месенджеру телеграм;
- користуючись отриманою інформацією, дослідити методи підвищення захищеності месенджеру;
- реалізувати прототип програмного забезпечення для реалізації запропонованого методу.

Актуальність дослідження полягає в необхідності розвитку методів для підвищення захисту месенджерів, та їх практичної реалізації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз основних моделей зловмисників

В [1] було проведено ґрунтовне дослідження безпеки сучасного обміну повідомленнями, виходячи з отриманих результатів можна виокремити, що модель за якою представляється захист інформації несе в собі три види можливих зловмисників, що можуть нести небезпеку для інформації користувачів, їх класифікацію наведено нижче.

Глобальні – бувають активними та пасивними, ключовою їх рисою є те, що вони можуть отримати контроль над значною частиною інтернет послуг. Прикладом є великі інтернет-провайдери або відповідні структури могутніх держав.

Місцеві – можуть бути активними та пасивними, вони можуть приєднуються, та контролювати локальні мережі по обидві боки комунікації. Прикладом можуть бути користувачі (навіть самі власники) відкритих точок доступу до інтернету (в закладах харчування, громадських місцях чи аеропортах).

Також, як окремий вид зловмисників, визначають постачальників послуг. В разі, коли додатки, в своїй роботі, для розповсюдження відкритих ключів використовують централізовану інфраструктуру, усі оператори можуть бути розглянуті як потенційні злочинці.

Проаналізувавши офіційну документацію таких найрозповсюдженіших в світі месенджерів як: Viber [2], Facebook Messenger [3], Telegram [4] та WhatsApp [5], можна зробити висновок, що в кожному з них використовуються подібні методи захисту інформації. В офіційних документах заявляється, що кожен з месенджерів надає «надійну» безпеку для

даних їхніх користувачів. Виходячи з цього, повинен бути реалізований якісний захист від кожного з класифікованих вище можливих зловмисників.

1.2 Аналіз захищеності Telegram

Для реєстрації будь який користувач повинен мати номер телефону (незалежно від країни реєстрації), та можливість приймати на нього SMS. Реєструючись одного разу за допомогою мобільного телефону, користувач отримує можливість використовувати створений обліковий запис (акаунт) на необмеженій кількості пристроїв, як в мобільних застосунках, так і в десктоп клієнтах або веб-браузерах [4].

Після реєстрації, Telegram вимагає заповнити інформацію про себе [6].

Обов’язкові поля обмежені одним лиш ім’ям. Воно може містити будь яку інформацію, в тому числі будь який символ (навіть пробіл).

До не обов’язкових відноситься: фамілія, зображення профілю (є можливість додавати будь яку кількість фотографій), та інформація «про себе». Також після реєстрації є можливість створити ім'я користувача (не плутати з ім’ям) або його ще називають «username». Воно буде містити символ «@» на початку та повинно бути унікальним, тому що має точно ідентифікувати обліковий засіб користувача. Користувацькі налаштування дають можливість закрити відображення кожного з цих полів (в тому числі номер телефону) окремим групам користувачів (або конкретним користувачам) [6]. Наприклад, показувати фото профілю контактам і не показувати всім іншим.

Таким чином, аналізуючи можливості налаштування та політику конфіденційності [6] можна зробити висновок, що Telegram дає можливість та інструменти для створення облікового запису з високим рівнем конфіденційності.

З замовчуванням для спілкування Telegram пропонує звичайний (до яких також відносяться групові) та секретний чат. Можливість створювати та

підписуватися на Telegram канали. Також, в месенджері можна здійснювати голосові та відеодзвінки. Є можливість передавати зображення, аудіо, відео та інші файли. Записувати голосові та відео повідомлення. Крім тексту в повідомленнях є можливість використовувати статичні emoji – смайлики, та стікери. В 2022 в Telegram стала доступна функція групових відео трансляцій (стрімів).

Месенджер Telegram є безкоштовним, але 7 листопада 2021 був зроблений крок до монетизації, та з'явилася перша реклама, що розміщується в вигляді одного повідомлення в низу стрічки будь якого з каналів, сумарна аудиторія яких більше тисячі користувачів. Хоч така реклама може розміщуватися тільки через офіційний сервіс Telegram Ads, значна кількість користувачів скаржаться на низький рівень якості оголошень, та іноді відверто шахрайський контент. В 2022 Telegram впровадив платну підписку Premium. Хоч, вона офіційно не надає ніяких переваг в захищеності, але дає ряд можливостей, що значно покращують досвід використання, та опосередковано все-таки впливають на безпеку користувачів. Розширена версія, дозволяє використовувати анімовані смайлики, статуси, зображення профіля, підвищення швидкості завантаження файлів, та їх максимального розміру, дає можливість записувати «відео-історії», перетворювати вхідні аудіо повідомлення в текст, та що головне, відключає показ реклами. Останнє огороджує користувачів від можливих шахраїв, що діють через фішинг або соціальну інженерію.

Більшість користувачів використовують саме звичайні особисті чати, рідше є учасниками групових чатів, та ще рідше використовують секретні чати.

Звичайні (хмарні) чати Telegram - доступні будь яким користувачам за замовчуванням, важливим є те, що вони не забезпечені наскрізним шифруванням. Це означає, що всі повідомлення, будь який медіа контент і документи в таких листуваннях зберігаються на серверах Telegram і доступні

з декількох пристроїв [6]. Для пересічного користувача це несе вагому зручність використання, але додає питань, та можливих причин для занепокоєння з приводу безпеки та конфіденційності користувацьких даних. Telegram офіційно стверджує, що дані користувачів зберігаються на серверах в зашифрованому вигляді, а ключі шифрування не зберігаються в одному дата-центрі. Також, для ускладнення фізичного доступу до даних користувачів, місцевими урядами (без законної підстави) або зловмисниками (в тому числі технічного персоналу) ключі розосереджені в різних юрисдикціях [6].

Секретні чати Telegram – напроти, в своїй основі передбачають використання наскрізного шифрування, автоматичне самознищення повідомлень (з можливістю налаштування таймеру), а також відсутність всієї інформації чату в хмарі [6]. Доступ до конкретного секретного чату можливий лише з девайсів на яких почалося таке листування. Такий підхід забезпечує ще один рівень безпеки та конфіденційності розмов та запобіганню доступу (перехвату або перегляду) до змісту повідомлень нікому крім авторизованого користувача та його співрозмовника.

Відео та аудіо дзвінки в Telegram – це досить зручні функції які також повинні забезпечувати достатній рівень безпеки та конфіденційності. Аналізуючи [4] та [7] можна сказати що всі види дзвінків захищені наскрізним шифруванням. Це забезпечує високий рівень конфіденційності і безпеки для інформації, що передається користувачами.

Політика конфіденційності Telegram гарантує, що конфіденційні користувацькі дані (в тому числі що передаються через дзвінки) не використовуються для показу реклами. Рекомендація рекламного контенту побудована на загальній інформації про користувача, такій як префікс телефонного номеру та направленість інтересів (виходячи з підписок на публічні канали)[6].

В матеріалах присвячених 16-тій конференції міжнародної асоціації обчислювальної техніки (ACM - Association for Computing Machinery) про безпеку та конфіденційність у бездротових та мобільних мережах, йдеться про те що основними шляхами захисту інформації користувачів месенджерах, в тому числі і в Telegram, є використання технологій наскрізного шифрування, шифрування даних на серверах та зобов'язання користувачів застосовувати двофакторну аутентифікацію [8].

Наскрізне шифрування (End-to-End Encryption -E2EE) – це метод захисту даних листування, при якому ніхто крім відправника та одержувача повідомлення не має доступу до вмісту.

Процедура обміну повідомленнями з використанням наскрізного шифрування передбачає таку послідовність дій:

1) вміст повідомлення (текст, зображення, документи, аудіофайли) шифруються на девайсі (тобто, на смартфоні або комп'ютері) користувача;
2) в зашифрованому вигляді відправляються отримувачу;
3) тільки після отримання користувачем, повідомлення розшифровується.

При такому алогритмі дій, ніяка третя особа, яку можна було представити за класифікацією наведеною вище, не може отримати доступу до змісту повідомлення. Тобто, при передачі, дані в мережі шифруються, але сам факт такої передачі залишається видимим [9].

Виходячи з [8], в порівнянні з іншими месенджерами, які переважно використовують Signal протокол, в Telegram використовується власний протокол шифрування MTProto. Відповідно до документації Telegram, протокол призначений тільки для доступу до серверного API з додатків, працюючих на мобільних пристроях. Важливо зазначити, що він не призначений для будь-яких веб браузерів та десктопних клієнтів. В разі, коли Telegram’ом користуються в клієнті, встановленому на комп’ютері, чи в веб-

браузері - використовується один з стандартних протоколів, що зазвичай використовуються для безпечного зв’язку через інтернет. Найчастіше це HTTPS (HyperText Transfer Protocol Secure) або TLS (Transport Layer Security). Важливо відмітити, що Telegram також пропонує проксі сервіс під назвою MTProto Proxy, який дозволяє користувачам маршрутизувати свій трафік через проксі-сервер з використанням MTProto.

Протокол MTProto повинен забезпечувати надійність доставки повідомлень при поганій якості інтернету та безпечну передачу даних при швидкій передачі [4]. Поточна версія протокола застосовується як для клієнт-серверного шифрування так і для наскрізного шифрування. MTProto складається з трьох частин, які майже не залежать одне від одної.

1) Високорівнева частина – відповідає за перетворення запитів до API та відповідей від нього в двійковий код.
2) Криптографічна (авторизаційна) частина – відповідає за шифрування перед передачею через транспортний протокол.
3) Транспортна частина – відповідає за організацію передачі повідомлень поверх інших мережевих протоколів (таких як, https, tcp, udp) [4].

В Telegram функція наскрізного шифрування, не являється обов’язковою, тому що доступна лише в секретних чатах та дзвінках[10].

Чинна версія MTProto – 2.0, була введена у використання на версії Telegram 4.6. Минула версія (MTProto 1.0) зараз не використовується зовсім через свою застарілість[4]. Не дивлячись на те, що MTProto піддавався критиці [11] в ньому були знайдені вразливості, які навіть були занесені до бази CVE (Common Vulnerabilities and Exposures) [12], Telegram дуже оперативно випускав оновлення та вирішував проблеми з безпекою, та після виконаної роботи навіть випускав докази надійності протоколів для автентифікації, звичайного чату, наскрізного шифрування і зміни ключів [7]

Двофакторна перевірка або також відома як двофакторна аутентифікація (2FA - two-factor authentication) – ще одна технологія, за допомогою якої досягається підняття рівня захищеності в Telegram. Для того щоб увійти в Telegram з нового пристрою користувач повинен ввести SMS-код який посилається на телефон або в уже розпочату сесію на іншому девайсі. Якщо ж юзер активує, в налаштуваннях конфіденційності, функцію 2FA крім коду перевірки він повинен буде ввести додатковий пароль (задається одноразово при активації функції) [4].

Таким чином використання функції 2FA – це доволі простий і зрозумілий спосіб обмежити доступ до облікового засобу можливим зловмисникам, навіть якщо номер мобільного телефону на який зареєстрований Telegram аккаунт потрапив у руки комусь іншому. Головна порада під час налаштування 2FA - проявляти обережність та задавати надійний пароль.

Ще одним недоліком в захищеності користувацьких даних є те, що якщо виникає необхідність передати секретне повідомлення саме через груповий чат, так щоб його зміст був доступний тільки конкретним адресатам, Telegram не передбачає для цього інструментів.

Підсумовуючи можна сказати, що Telegram дійсно надає високий рівень безпеки даних користувачів. Надає можливість робити гнучкі налаштування безпеки, гарантує конфіденційність та надає інструменти для її забезпечення. Найголовнішими технологіями для цього є наскрізне шифрування, шифрування на серверах та функція 2FA. Використання всього переліку функцій в теорії повинне забезпечити безпеку від основних моделей зловмисників. Не дивлячись на все це, є й слабкі місця. На нашу думку, відсутність відкритого коду, як серверної так і клієнтської частини, може нести в собі приховані ризики. Той факт, що наскрізне шифрування не є обов’язковим, тобто використовується в опортуністичному режимі являється конкретним недоліком безпеки. Ще один з недоліків полягає в тому, що E2EE

не використовується в групових або корпоративних чатах. Виходячи з аналізу [8] поточні реалізації функції E2EE в різних месенджерах, особливо в опортуністичному режимі, можуть перемогти пасивного man-in-the-middle (MitM)-атакувальника, але не завжди може перемогти активного. Таким чином, хмарні чати мають меньший рівень конфіденційності і потребують додаткових заходів безпеки. Також, якщо користувачі повністю довіряють, що сам Telegram не збирає та не аналізує конфіденційну інформацію, потрібне рішення яке дасть змогу передати повідомлення (наприклад, пароль) більш захищеним способом так, щоб тільки конкретний користувач мав до нього доступ, навіть у групових чатах.

1.3 Аналіз можливих підходів підвищення захищеності Telegram

Спираючись на проведений аналіз захищеності Telegram, можна сказати що наявні слабкі місця. Зокрема особисті та публічні чати в порівнянні з секретними мають менший рівень захищеності.

Загалом, системи інформаційної безпеки можна поділити на два класи - скривання інформації і її шифрування (шифрування розглядається в контексті криптографії) [13]. Вони відрізняються підходами, але при правильній реалізації, за допомогою кожного з них можна успішно захищати дані.

Криптографія - це наука що вивчає та досліджує методи забезпечення цілісності, неспростовності, конфіденційності та автентифікації даних [13].

Також, криптографія розглядається в якості однієї з фундаментальних технік захисту інформації та зв'язку, яка реалізується за допомогою математичних концепцій і алгоритмів для перетворення даних у набір символів що важко розшифрувати [14].

Іншою фундаментальною технікою для захисту інформації є стеганографія. На відміну від криптографії, вона заснована на ідеї скрити сам факт наявності вбудованого повідомлення. Можна сказати, що стеганографія – це мистецтво приховування секретної інформації в загальнодоступних не

секретних матеріалах. Про наявність там конфіденційної інформації повинні знати тільки відправник та отримувач [9].

Важливо зазначити, що на конференції Information Hiding (First Information Workschop) у 1996 році були обговорені всі базові поняття стеганографії та прийнята єдина термінологія[15][16]. Згідно з цією термінологією стеганографічна система або стегосистема – це сукупність засобів та методів, які використовуються для формулювання таємного каналу зв'язку. Будь-яка інформація, в якій приховані таємні дані, називається контейнером. За контейнер може слугувати будь-який файл чи потік даних. Контейнер, який не містить таємного повідомлення, називають порожнім, а той, що містить – заповненим або стегоконтейнером. Канал передачі сгегоконтейнера має назву стеганографічного каналу або стегоканалу. Таємний ключ, який необхідний для «вкраплення» інформації в контейнер, називається стегоключем або просто ключем. Залежно від кількості рівнів захисту в стегосистемі може використовуватись як один, так і декілька ключів[17]. Процес виявлення вбудованих даних називають стеганоаналізом [18].

В процесі аналізу цих, різних в своїй сутності, підходів було розроблено порівняльну таблицю 1.1.

Таблиця 1.1 - Порівняння стеганографії та криптографії

Аспект	Стеганографія	Криптографія
Принцип	Приховування існування повідомлення	Трансформація змісту повідомлення
Мета	Невідомість і невиявленість	Безпека вмісту та нечитабельність

Продовження таблиці 1.1 - Порівняння стеганографії та криптографії

Фокус	Приховування того факту, що відбувається передача повідомлення	Забезпечення того , що повідомлення є незрозумілим для неавторизованих осіб
Метод	Вбудовування інформації в несекретний носій	Перетворення початкового тексту в зашифрований текст
Сфера застосування	Приховане спілкування, водяні знаки	Безпечний зв'язок, цілісність даних, автентифікація
Забезпечення захисту	Залежить від секретності техніки	Залежить від потужності алгоритму та секретності ключа
Ризик виявлення	Низький, поки факт вбудування не буде виявлено	Високий, але вміст залишається захищеним
Використання ресурсів	Зазвичай низьке, обмежений розміром носія	Може бути високим, особливо з надійними методами шифрування
Обмеження	Ефективність використання знижується, якщо метод відомий	Вимагає керування ключами, ресурсомісткий

Провівши порівняльний аналіз, можна сказати що використання стеганографії буде більш доречним в сценаріях, де секретність має першочергове значення. З іншого боку, криптографія зосереджена на захисті вмісту повідомлення, що робить її ідеальною для забезпечення

конфіденційності та цілісності даних що передаються. Обидва розглянуті підходи мають унікальні переваги та обмеження і їх використання залежить від конкретних вимог сценарію безпеки. Але факт того, криптографія змінює зовнішній вид повідомлення що передається, робить таке повідомлення більш цікавим для можливих зловмисників. Не дивлячись на те, що за допомогою криптографії потенційно можна захистити більший обсяг інформації та такий захист буде більш стійким, для підвищення захищеності Telegram є сенс використовувати саме стеганографічні алгоритми. Такий підхід не є розповсюдженим та при правильній реалізації може забезпечити гарний рівень захисту даних. Виходячи з цього є необхідність проаналізувати стеганографічні алгоритми та використовувати такі що мають мають гарну пропускну здатність та при цьому показують мінімальну зміну стегоконтейнеру.

1.4 Аналіз найбільш придатних типів стегано контейнерів

Стеганографія, за допомогою якої відбувається скривання даних в цифрових об’єктах будь яких видів називається цифровою [19].

В цифровій стеганографії розрізняють 2 основні напрями:

1) Такі, що не передбачають цифрову обробку сигналів (ЦОС).
2) Та такі, що для вбудовування конфіденційної інформації використовують ЦОС [20].

В першому випадку стеганорафічним контейнером є заголовки файлів або пакетів даних. Це робить ідентифікування, вилучення та знищення вбудованої інформації досить легким, і тому такий метод не знайшов широкого застосування [19].

Стеганографія з використанням ЦОС дає можливість використовувати в якості стегоконтейнерів текст, аудіо, відео, інші медіафайли та навіть мережеві

протоколи [20]. Останній є досить складним в реалізації тому не буде розглянутий.

Узагальнюючи можна сказати, що залежно від носія, який використовується для вбудовування даних, стеганографія може бути класифікована як текстова стеганографія, стеганографія зображень (графічна стеганографія) та аудіо стеганографія [21].

Для реалізації текстової стеганографії проводиться модифікація атрибутів тексту. В якості таких атрибутів може використовуватися символи((в тому числі що не відображаються) або навіть орфографічні помилки. В загальному розрізняють три види вбудовування в текст: заснований на статистичній або випадковій генерації, на лінгвістичних надмірності та маніпулюванні форматуванням. Текстова стеганографія має ряд суттєвих недоліків, через що вона майже не використовується. Найбільш суттєвий, полягає в тому, що текст має меньше надмірних даних, що в порівнянні з зображеннями чи аудіофайлами. Це впливає на кількість можливої інформації, що може бути вбудована без серйозного впливу на стегоконтейнер [27].

У графічній стеганографії для приховування повідомлення, яке має бути передане таємно, використовується графічний сигнал, тобто майже будь які формати файлів зображень які мають надлишковість для вбудовування туди прихованої інформації [22]. Вбудовування в відеофайли теж відноситься до стеганографії зображень, тому що відео представляє собою зміну кадрів, кожен з яких є зображенням. Графічна стеганографія є однією з найбільш використовуваних, через відносну легкість реалізації та зрозумілість логіки роботи.

Аудіо стеганографія у якості стеганоконтейнеру розглядає оцифрований аудіосигнал. Аудіосигнали по своїй природі мають надмірну надлишковість. Наявність природних шумів, зазвичай сам по собі великий розмір та висока

швидкість передачі даних дає можливість вбудовувати більший відсоток інформації в порівнянні з контейнерами інших типів.

Важливим є те, що в загальному стеганографія в якій в якості стегоконтейнеру використовується зображення або відео грунтується на обмеженнях зорової системи людини (ЗСЛ), в той час як аудіо стеганографія на обмеженнях слухової системи людини (ССЛ). Зрозуміло що поняття чутливості є досить індивідуальним і може змінюватися від людини до людини, але зазвичай ССЛ менш чутлива до мінімальних змін ніж ЗСЛ.

Також потрібно зазначити, що за допомогою стеганографічних методів , в зазначені вище типи контейнерів, може проводитися вбудовування інформації не тільки в виді тексту. Це можуть бути , наприклад, приховування зображень в аудіо [23] [30], аудіо файлів в зображеннях [24], відео в аудіо [32] та інші різні комбінації застосування стеганографічних методів .

Проаналізувавши наявні види стегоконтейнерів, вибір було зроблено в користь аудіофайлів. Ключовими в цьому стали ряд характеристик:

- надмірна надлишковість інформації (наявність природного шуму робить аудіо стеганографію більш привабливою, ніж стеганографію тексту та зображень);

- менша чутливість слухової системи людини до мінімальних змін (індивідуально, але зазвичай менша, ніж для зорової системи, слабку чутливість до змін якої використовують в стеганографії зображень);

- широка доступність та використання аудіофайлів.

В цьому контексті є необхідність проаналізувати наявні методи аудіо стеганографії.

1.4.1 Порівняльний аналіз основних методів аудіо стеганографії

Не дивлячись на широке коло стеганографічних систем, вони мають однакові основоположні аспекти [26]:

Міцність (robustness) – ця характеристика визначає наскільки система здатна витримувати різні атаки, тобто забезпечувати після них зберігання цілісності, незмінності та не ідентифікованості вбудованого повідомлення[26];

Вмістимість (capacity) або пропускна здатність – характеризує максимальну кількість інформації, яка може бути вбудована в стеганоконтейнер так, щоб не викликати його зміну для спостерігача та не збільшити можливість появи помилок під час передачі через канал зв'язку[27];

Не змінність (transparency) або відсутність змін контейнеру після вбудування – характеризує відсутність видимої модифікації або спотворення стегано контейнеру після вбудовування в нього інформації[26].

Існує безліч стеганографічних методів, які в якості стегоконтейнерів використовують аудіофайли. З метою здійснення порівняльного аналізу [28] таких з них, як метод кодування найменш значущого біта (LSB) [20], методу фазового кодування (Phase coding), розширеного спектру (Spread Spectrum) та приховування відлуння-сигналу (Echo Hiding). Були проаналізовані такі характеристики як властивості слухової системи людини, що використовуються в кожному з розглянутих методів, особливості аудіо контейнерів в які вбудовується повідомлення, теоретичну пропускну здатність, стійкість до стегано атак, теоретичну відсутність спотворення стегано контейнеру після вбудовування в нього інформації, складність реалізації та складність обчислення, було розроблено таблицю 1.2:

Таблиця 1.2 - Результат порівняльного аналізу основних стеганографічних методів

Критерії порівняння	Метод LSB	Метод фазового кодування	Метод розширеного спектру	Метод приховування відлуння
Властивості ССЛ	Слабка чутливість ССЛ до незначної зміни гучності аудіосигналу	Несприйнятливість ССЛ абсолютної фази аудіосигналу	Нездатність ССЛ сприймати невеликі зміни в широкому діапазоні частот	Неможливо легко розрізнити штучне відлуння за певних порогових значеннях
Особливості Аудіо Контейнерів	Найкраще для сигналів з високою бітовою глибиною	Підходить для складних музичних треків	Широкий діапазон, особливо з широким спектром частот	Потрібен достатній динамічний діапазон для маскування відлуння
Теоретична пропускна здатність	Висока (1 Кб/сек на 1 кГц в каналі без перешкод)	Нижче ніж LSB	Помірний	Від низького до середнього

Продовження таблиці 1.2 - Результат порівняльного аналізу основних стеганографічних методів

Стійкість до стегано атак	Низький	Помірний	Високий	Помірний
Теоретична відсутність спотворень	Мінімальна, але не відсутній	Відносно висока	Висока	Помірна
Складність реалізації	Низький	Помірний	Високий	Помірний
Складність обчислень	Низький	Помірний	Високий	Помірний

Тобто, Метод LSB є простим в реалізації та має гарну пропускну здатність через вбудовування інформації в кожен останній найменш значущий біт, але вагомими недостатками є низька стійкість до статистичного аналізу та модифікації стегоконтейнеру. Також недоліком є утворення відчутного на слух низькочастотного шуму. Метод розширеного спектру забезпечує високу стійкість до більшості форм обробки сигналу, включаючи стиснення та повторне кодування. Його розсіяний характер по частотному спектру робить його менш чутливим до цілеспрямованих змін. Також, є досить стійким до виявлення присутності закодованих даних, та потребує вдосконалених інструментів стегоаналізу. Метод приховування даних з використанням відлуння сигналу має досить високу стійкість до несанкціонованого детектування та вилучення інформації та гарне співвідношення інших характеристик, але істотною проблемою є те, що метод погано досліджений. Як результат метод не допрацьований до промислового використання. Тому фаворитом для використання є метод фазового кодування. Метод є одним з найбільш ефективних методів за критерієм співвідношення сигнал/шум, що

сприймається ССЛ. Він хоч і має теоретично меншу відсутність спотворень та меншу стійкість до стегано атак ніж метод розширеного спектру, зате є значно простішим в реалізації. В цьому контексті є необхідність розглянути реалізацію стеганографічного методу фазового кодування більш глибоко.

1.4.2 Аналіз стеганографічного методу фазового кодування

Метод фазового кодування спрямований на вбудовування даних непомітним для людського вуха способом шляхом маніпулювання фазовими компонентами спектра аудіосигналу [39]. Він ефективний з точки зору пікового співвідношення сигнал/шум (PSNR), яке вимірює якість реконструйованого аудіосигналу після вбудовування прихованих даних [39].

Логіка роботи

Метод фазового кодування пропонує використання перетворення Фур'є і кодування початкових фаз сигнала в частотній області. Логіка виконання має такий порядок кроків:

1) Вхідний аудіо сигнал розбивається на блоки однакової довжини.
2) Для кожного блока виконується дискретне перетворення Фур'є (ДПФ). Воно перетворює часову функцію в коєфіцієнт Фур’є та дає 3 масиви значень: масив амплітуд , частот та фаз.
3) Запам'ятовується різниця фаз між кожними сусідніми сегментами, тобто обраховується масив різниць фаз.
4) Двійкова послідовність даних кодується у масив початкових фаз
5) Ці значення поміщаються замість будь якого масиву початкових фаз одного з блоків
6) Використовуючи масиви амплітуд, фаз і модифікованого масиву частот виконується зворотне ДПФ для кожного блока.

Так як блоки оброблялися незалежно один від одного, в місцях з'єднання блоків можуть з’явитися розриви (різкі перепади). Ця ситуація буде виявлена. Для усунення цих ефектів можна виконати згладжування (апроксимацію). Такі

процеси спотворюють в частотній області значення початкових фаз, але не повинні призвести до втрати інформації.

1.4.3 Аналіз форматів аудіо стегоконтейнерів

Через те що кількість інформації, яка вбудовується, напряму залежить від характеристик стегоконтейнеру, в контексті аудіо стеганографії є актуальним аналіз типів аудіофайлів. Проаналізувавши їх характеристики можна буде аргументовано робити між ними вибір.

Найчастіше використовуються такі формати аудіофайлів, як WAV (формат аудіофайлу Waveform), MP3 (MPEG-1 Audio Layer III), AAC (розширене кодування звуку), OGG (Ogg Vorbis), AIFF (формат файлу обміну аудіо).

Для змоги більш якісно проаналізувати типи аудіофайлів, було розроблено таблицю 1.3.

Таблиця 1.3 - Основні характеристики аудіо файлів в контексті стеганографії

Аудіоформат	характеристики	Придатність для вбудовування даних	Недоліки
WAV	Якісний звук без стиснення	Відмінно, легко маніпулювати, без втрати якості.	Великі розміри файлів
MP3	Стиснений, із втратами, досить популяриний	Складне стиснення з втратами впливає на приховані дані	Стиснення може перешкодити прихованим даним

Продовження таблиці 1.3 - Основні характеристики аудіо файлів в контексті стеганографії

FLAC	Стиснений, без втрат, зберігає оригінальний звук	Гарне стиснення без втрат зберігає дані без змін	Більший за формати з втратами даних, менший за WAV
AAC	Ефективне стиснення, вища якість, ніж MP3	Проблеми, схожі на MP3, ефективні за нижчих бітрейтів	Стиснення з втратами може вплинути на приховані дані
OGG	Стиснений, із втратами, має відкритий код	Стиснений, із втратами	Стиснення з втратами може вплинути на приховані дані
AIFF	Схожий на WAV, без стиснення, високої якості	Дуже добре, схоже на WAV	Великі розміри файлів

Вибір типу аудіофайлу, для використання його в якості стегоконтейнеру, залежить від багатьох факторів. Виходячи з того, що основна задача будь якої розробленої стеганосистеми скрити факт передачі секретного вбудованої інформації, стегоконтейнер повинен не тільки мати гарні характеристики, а й не виділятися на фоні інших файлів. Виходячи з таких міркувань, потрібно використовувати популярні типи аудіофайлів. MP3 - найвідоміший з перерахованих вище, але той факт, що він має стиснення з втратами, робить його мало корисним. WAV формат найкращий для заявленої вище цілі. Він не використовує стиснення та має звук високої якості. Також він є досить відомим, тому не повинен привертати зайвої уваги. Інші розглянуті формати

хоч в деяких випадках і мають гарні характеристики (AIFF, FLAC), на жаль є менш розповсюдженими.

1.5 Чат-бот як механізм підвищення захищеності Telegram

Додавання забезпечення захисту інформації, за допомогою будь якого з цих підходів, утворює додатковий крок у спілкуванні користувачів. Якщо користувач свідомо на це йде , то основною задачею стає саме забезпечення підвищення захищеності його спілкування. Тобто для широкого користувача не повинно мати значення як саме це реалізується, головне щоб була досягнута мета та сам процес був досить простим. В цьому контексті було запропоновано реалізувати процес стеганографічних перетворень за допомогою телеграм боту. Telegram боти, як звична складова простору Telegram месенджеру зроблять досвід користувача максимально “безшовним”. Тобто не потрібно буде використовувати інші сервіси, завантажувати та налаштовувати додаткове ПЗ та навіть відволікатися від інтерфейсу самого Telegram.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Після того як досліджена захищеність Telegram, проведений аналіз його слабких місць та у якості найкращого рішення обрано стеганографічні алгоритми, тобто доведена необхідність виконання цієї роботи, - потрібно виконати проектування програмного забезпечення (ПЗ), що представляє собою Telegram бот.

2.1 Розробка концептуальної моделі програмного забезпечення

Взявши за основу методологію розробки програмного забезпечення Rational Unified Process (RUP), для початку було необхідно побудувати концептуальну модель, а вже на основі неї було зроблено опис роботи ПЗ. Щоб задовольняти предметну область, така модель має враховувати в себе інтереси всіх учасників системних процесів.

Головною ідеєю RUP – є розбиття всього процесу розробки ПЗ на логічні етапи. Такий підхід забезпечує змогу адаптуватись до зміни потреб проекту, та передбачає використання візуального моделювання через уніфіковану мову моделювання (UML). Він забезпечує систематичний підхід для охоплення всіх фаз життєвого циклу розробки програмного забезпечення (SDLC).

Важливо зазначити, що в концепції RUP концептуальна модель функціонування ПЗ та її операційне оточення визначається можливими варіантами використання (прецедентами).

Першим кроком було виокремлено акторів, тобто сутностей що будуть взаємодіяти з програмним забезпеченням:

- Адміністратор боту - фахівець, що в разі необхідності відстежує та виправляє помилки, має доступ до всіх функцій.

- Користувач – будь які користувачі Telegram, які хочуть вбудувати або витягти (в цьому випадку вони повинні знати довжину сегменту) дані з аудіо файлу формату WAV.

Також були виокремлені та представлені у виді таблиці основні сценарії, що описують використання ПЗ певним актором для вирішення однієї із задач (тобто прецеденти):

Таблиця 2.1 - Основні прецеденти

№	Прецедент	Актор	Порядок дій
		Користувач	• Запустити бота; • натиснути кнопку “Приховати дані в аудіо”; • Відправити необхідний аудіо файл формату WAV; • Відправити необхідний текст для вбудовування • Запам’ятати ширину сегменту; • Отримати зашифрований файл; • Повернутися в меню.
		Адміністратор	• При появі помилок в логу боту вирішити проблему.

Продовження таблиці 2.1 - Основні прецеденти

2	Вилучення вбудованих даних з аудіо файлу формату WAV	Користувач	• Запустити бота; • натиснути кнопку “Вилучити дані”; • Відправити необхідний аудіо файл формату WAV; • Відправити ширину сегменту; • Отримати вилучений текст; • Повернутися в меню.
		Адміністратор	• При появі помилок в логу боту вирішити проблему.

Щоб краще проаналізувати взаємодію акторів з ПЗ було розроблено UML діаграму прецедентів (рисунок 2.1)

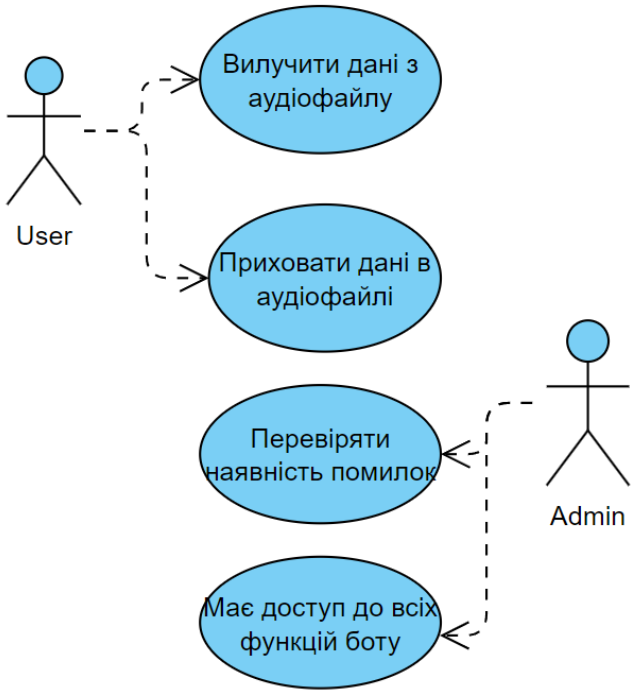

Рисунок 2.1- UML діаграма прецедентів

Процес відповіді клієнта на відправку запиту було представлено в вигляді діаграм діяльності на рисунку 2.2.

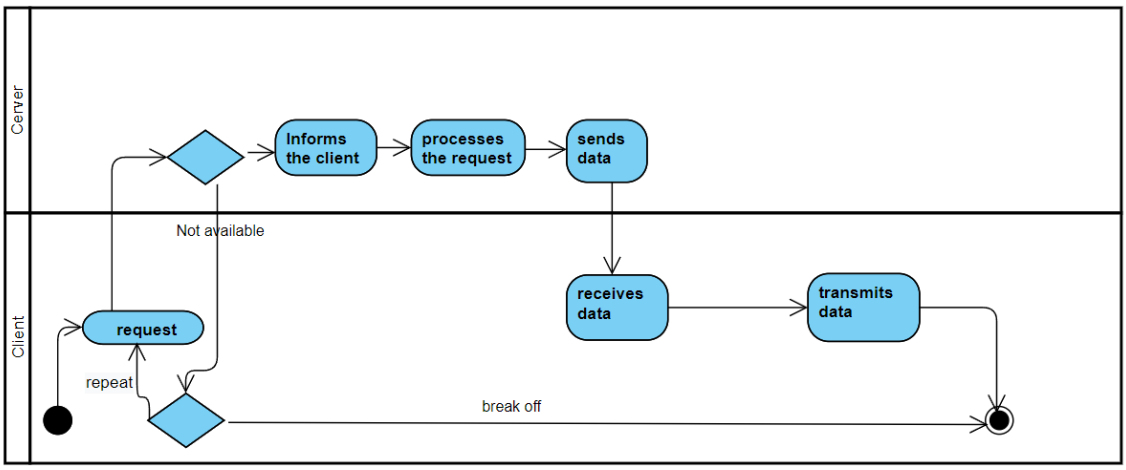

Рисунок 2.2 Діаграма діяльності

Діаграми активності використовують для розробки моделей робочих процесів у програмних системах (ПС). За допомогою них можна краще

проілюструвати які дії виконує система. Вони показують послідовність кроків, розгалуження, паралельність і загальний потік керування.

2.2 Проектування патерну системної архітектури

Вибір оптимальної архітектури це один з основних етапів розробки будь якого програмного забезпечення.

В проектуванні програмного забезпечення архітектурні шаблони є загальними рекомендаціями для недопущення типових проблем у структурі розроблюваних систем [32]. Вони допомагають архітекторам та розробникам приймати обґрунтовані рішення щодо організації своїх проектів, покращуючи якість, зручність обслуговування та масштабованості [33].

Застосування готових шаблонних рішень дозволяє розробникам сконцентруватися на унікальних вимогах свого проекту. Це сприяє створенню більш зручної та масштабованої програмної системи, а також зменшує можливі ризики при впровадженні нових функцій чи обробці збільшених навантажень користувачів [32].

Зазвичай, чат-боти мають архітектуру що складається з:

- Клієнтської частини – складова ПЗ, що з якою безпосередньо взаємодіє користувач;
- Серверної частини – складова ПЗ, де реалізована вся логіка для обробки команд користувачів.

В залежності від поставленої мети, різні архітектурні шаблони можна узагальнити, розбивши їх по відповідним групам. За допомогою них можна легко зрозуміти як влаштований продукт всередині і як в загальному працює логіка. Такі групи називаються - паттернами системної архітектури. Для реалізації ПЗ було обрано клієнт-серверну архітектуру.

Клієнт серверна архітектура – це підхід який означає розподіл завдань між програмним забезпеченням на серверному комп'ютері та клієнтському

пристрої. Клієнтський комп'ютер звертається до сервера, викликаючи якусь команду або запитуючи інформацію [21]. Усе обчислювальне навантаження для опрацювання логіки та будь-яких розрахунків падає саме на сервер. Велика кількість підключених клієнтів тільки викликають необхідні дії або отримують потрібну відповідь, тому можуть мати слабкі технічні характеристики. Логіка роботи клієнт-серверної архітектури зображена на рисунку 2.3.

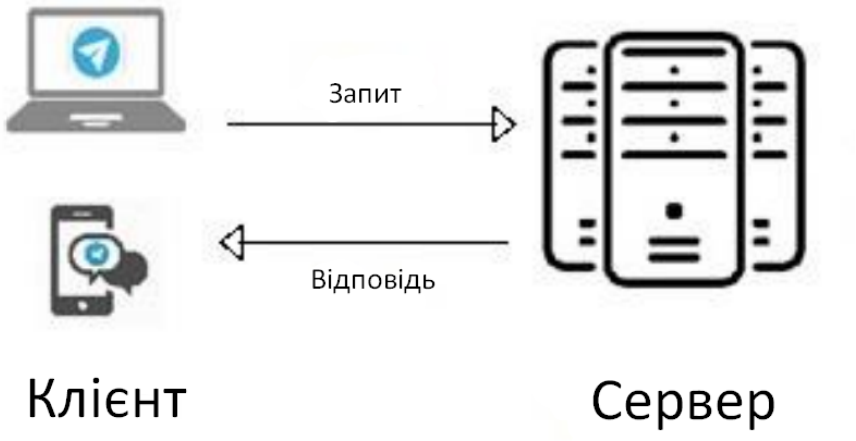

Рисунок 2.3 - Клієнт-серверна архітектура

Вибір клієнт-серверної архітектури обумовлений рядом переваг, основні з них перераховано нище.

a. Просте розгортання та обслуговування є основними перевагами патерну клієнт-сервер. Ролі розподілені між кількома автономними системами; отже, обслуговування клієнтів є незалежним.
b. Архітектура клієнт-сервер сприяє підвищеній маштабованості. Масштабованість програми є важливою перевагою, враховуючи останні тенденції розробки додатків, оскільки попит на технологічне вдосконалення стрімко зростає.

c. За рахунок централізованого зберігання даних, простіше організувати логіку отримки декількома клієнтами доступ із різних місць. Доступ до спільних ресурсів і служб полегшує керування, зміну та повторне використання програмних модулів [35].

Не дивлячись на вагомі переваги, клієнт-серверний патерн має ряд недоліків, які необхідно враховувати при обиранні її в якості архітектури в розроблюваному ПЗ. Основні недостатки та обґрунтування їх не значного впливу на розроблену ПЗ наведено нижче.

Єдина точка відмови: серверна частина є слабким місцем, тобто якщо відбудеться відмова системи, клієнти не зможуть отримати до неї доступ та під питанням стає збереженість даних. Цей недолік робить систему вразливою до збоїв і вимагає надійних механізмів резервного копіювання та відновлення.

Ризики для безпеки: якщо в ПЗ передбачено зберігання конфіденційної інформації, зазвичай вона знаходиться в базах даних (БД), це створює додаткові ризики безпеки. Зловмисники можуть вживати дій для зруйнування модифікації чи крадіжки таких даних.

Перевантаження системи: велика кількість клієнтських запитів може призвести до перевантаженості мережевого трафіку, сповільнюючи зв’язок між клієнтами та сервером і впливаючи на загальну продуктивність системи. Тому логіку роботи системи потрібно розробляти приділяючи особливу увагу продуктивності.

Тобто, деякі особливості архітектурного рішення можуть виступати, як в ролі переваг так і недостатків, все залежить від конкретної реалізації. Так як в якості патерну системної архітектури було обрано клієнт-серверну архітектуру, ПЗ необхідно розробляти враховуючи її особливості.

2.3 Розробка специфікації системних вимог

В цьому розділі було сформульовано основні вимоги до ПЗ

Вимоги зі сторони адміністратора ПЗ:

- У випадку появи помилок, лог на сервері повинен це відобразити;
- Лог повинен дозволяти чітко ідентифікувати помилку що виникла;
- Помилки повинні являти собою саме програмні збої, всі інші види помилок (в тому числі зі сторони користувача) повинні оброблятися логікою Telegram бота;
- Всі зміни в логіці ПЗ повинні вноситися лише адміністратором;
- ПЗ не повинен зберігати конфіденційних даних користувачів;
- ПЗ повинен бути кросплатформенним.
- Вимоги зі сторони користувача:
- ПЗ не повинно зберігати конфіденційних даних;
- ПЗ повинен надавати можливість вбудувати повідомлення в аудіофайл формату WAV за допомогою стеганографічних алгоритмів;
- ПЗ повинен надавати можливість вилучати вбудоване повідомлення з аудіо файлу формату WAV за допомогою стеганографічних алгоритмів;
- При спробі вбудувати повідомлення у аудіофайл іншого формату ПЗ повинно попередити про це користувача та попросити його спробувати ще раз;
- Всі нововведення повинні бути доступні користувачеві одразу ж після введення.

В результаті аналізу перелічених вимог був розроблений список функцій програми що розробляється:

- Можливість вбудовування інформації в аудіофайл формату wav;
- Можливість вилучення вбудованого повідомлення з аудіо файлу формату wav;

- Можливість обробки помилок.

2.4 Розробка архітектури програмного забезпечення

Обрана клієнт-серверна архітектура буде розроблена на основі компонентного стилю системної архітектури. За допомогою компонентно-орієнтованої архітектури (CBA) можна буде розділити систему на невеликі частини (компоненти), що розгортаються незалежно та мають можливість багатократно використовуватись. Це може призвести до скорочення часу що потрібен для розробки та тестування, а тому до підвищення надійності та гнучкості. При такому підході прогнозується, що заміна чи додавання нових компонентів частіше відбувається без серйозних збоїв [36/]

Результат розбиття програмної системи на окремі функціональні або логічні компоненти представляє собою чітко визначені комунікаційні інтерфейси, що містять методи, події та властивості. Таким чином CBA забезпечує вищий рівень абстракції та розділяє проблему на підпроблеми, кожна з яких пов’язана з розділами компонентів [37]

Однак, крім суттєвих переваг, деякі особливості CBA є справжніми недоліками, наприклад, необхідність розбивати програму на модульні та функціонально окремі частини. Це стає проблемою, коли програми досить великі. Крім того, ускладнюється контроль відповідності компонентів вимогам програми. Також, може ускладнюватись контроль оновлень та обслуговування версій бібліотек компонентів [36].

Згідно з документацією [38], є два метода за допомогою яких бот може обмінюватися повідомленнями з серверами Telegram API:

1) getUpdates – Pull: бот весь час відправляє запити серверу Telegram API, запитуючи чи не з’явилися нові повідомлення;

2) setWebhook - push: сервер Telegram API самостійно відправляє повідомлення боту, як тільки воно з’явиться.

Для реалізації Telegram боту було обрано перший варіант. Не дивлячись на те, що використання setWebhook виглядає більш доречним (в цьому випадку сервер повинен бути менш навантаженим), є ряд важливих недоліків які свідчать проти такого вибору. Основними з них є питання безпеки та необхідність непростого налаштування. Потрібно додатково розгортати веб-сервер, налаштовувати відповідні порти та SSL-сертифікат.

В результаті було отримано додаток, що періодично надсилає POST HTTP-запити до методу «getUpdates» Telegram API, продемонстровано на рисунку 2.4.

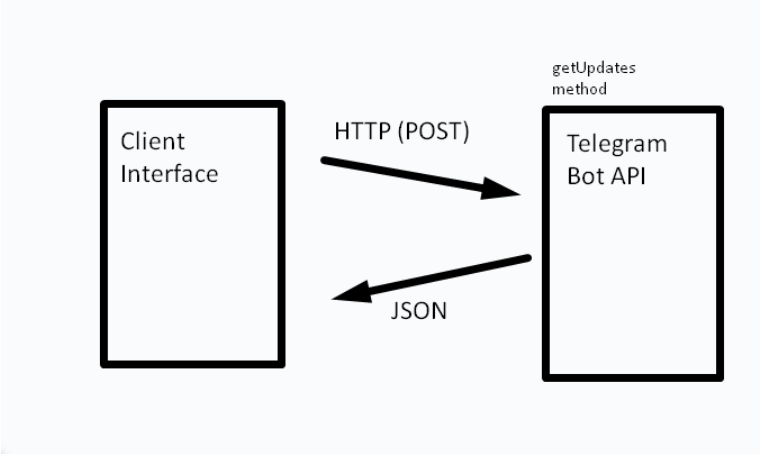

Рисунок 2.4. — Архітектура розробленого додатку

Тобто Telegram бот (клієнт і сервер в одному) надсилає POST HTTP-запити до методу «getUpdates» Telegram API. Так він перевіряє наявність нових повідомлень або команд, надісланих користувачами. Якщо з’явилися нові повідомлення, Telegram API відповідає повідомленнями у форматі JSON. Також важливо зазначити, що під час взаємодії з Telegram ботом користувачі не взаємодіють з серверною частиною напряму. HTTP- запити з клієнту надсилаються спочатку на Telegram API, а вже від нього на хостинг сервер боту.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В цій главі буде розглянутий весь процес розробки програмного забезпечення.

3.1 Вимоги до Telegram-бота

До розробленого Telegram-бота були виставлені основні вимоги:

Зручний і зрозумілий користувацький інтерфейс (простота взаємодії)

Користувач повинен мати можливість вирішити свої головні задачі за мінімальну кількість кроків. Зручний інтерфейс полегшує взаємодію з ботом для новачків. Швидкість навчання також росте, користувач може засвоїти логіку боту за менший час. Простота дизайну гарантує, що користування ботом буде доступне для широкої аудиторії, в тому числі для людей які не мають спеціальних знань. Простий у використанні інтерфейс робить його інклюзивним і привабливим для користувачів. Якщо користувачі вважають бота зручним і простим у використанні, вони з більшою ймовірністю продовжуватимуть взаємодіяти з ним протягом тривалого часу.

Таким чином розроблений інтерфейс боту, а саме прості та зрозумілі надписи кнопок, пояснень та логіка умовних дій має бути простою та інтуїтивно зрозумілим. Користувач не повинен заплутатися в спробах досягти своєї цілі – зашифрувати дані або виокремити з аудіозапису вже вбудовану інформацію.

Швидкість відповіді

По-перше, гарна швидкість роботи боту впливає на якість користувацького досвіду. Якщо Telegram-бот швидко обробляє команди, створюється враження про крашу взаємодію.

Здатність до масштабування

Ця риса є однією з головних коли йдеться про розробку рішення має бути конкурентним тривалий час. Врахування цього фактору, гарантує що при зростанні користувачів чат-боту, його теж можна буде масштабувати без шкоди для продуктивності. Здатність до легкого масштабування також гарантує, що можна буде включати нові функції чи проводити необхідні оновлення без додаткових витрат, пов’язаних із простоєм, надмірним виділенням ресурсів, або перебудовою спочатку розробленої логіки.

Надійність

Надійність – це дуже важливий фактор, на якому будується не тільки довіра користувачів, а й безпека конфіденційних даних. Коли надаються будь які користувацькі послуги, однієї помилки може бути достатньо, щоб підірвати довіру користувачів та змусити їх використати інші рішення. Практичні кроки для досягнення цього критерію складаються з:

- забезпечення резервного копіювання та відновлення у разі системних збоїв або сценаріїв втрати даних;
- надійний хостинг;
- реальне тестування;
- обробка помилок та збоїв.

Здатність обробки великих об'ємів даних

Проектування системи яка відповідає цьому критерію гарантує, що в разі різкого стрибка вхідних даних бот не «впаде», тобто не відбудеться відказу в обслуговуванні. Бажано не тільки забезпечити використання високо

продуктивне серверне обладнання, а й використовувати рішення на для оптимізації на рівні програмного коду.

3.2 Обрання технологій розробки

3.2.1 Аналіз платформи для розробки

Не дивлячись, на активне удосконалення мов програмування, сьогодні для будь якої специфічної задачі можна знайти свою мову програмування, яка підійде найкраще по ряду параметрів. Проведемо аналіз 3 мов програмування, що найчастіше використовуються для розробки Telegram ботів: Python, Ruby та JavaScript.

Python – це високорівнева мова програмування загального призначення, створена Гвідо ван Россумом і випущена в 1991 році [39]. Наразі це найпоширеніша багатоцільова мова програмування високого рівня, яка дозволяє програмувати в об'єктно-орієнтованій та процедурній парадигмах [40]. Python використовується для веб-розробки (на стороні сервера), розробки програмного забезпечення, математики, системних скриптів, машинного навчання, аналізу даних та в багатьох інших сферах [40][41].

Ruby – це інтерпретована високорівнева мова програмування, що підтримує функціональне, імперативне, об'єктно-орієнтоване та рефлексивне програмування [42]. Створений Юкіхіро «Мац» Мацумото, Ruby поєднує елементи з його улюблених мов, таких як Perl, Smalltalk, Eiffel, Ada та Lisp, щоб сформувати нову мову, яка врівноважує функціональне програмування з імперативним програмуванням [43]. Ruby відомий своєю зручністю використання, читабельністю та балансом між простотою та потужністю [43].

JavaScript (JS) – інтерпретована високорівнева мова програмування, що поряд з мовою розмітки гіпертексту (HTML) і каскадними таблицями стилів (CSS) є однією з основних технологій мережі Інтернет. Створена Бренданом

Айхом у 1995 році, вона використовується для створення настільних програм, мобільних додатків, серверної та веб-розробки [44].

Усі розглянуті мови програмування користуються попитом в конкретних галузях, та надають можливість виконати більшість конкретних задач. Галузі розробки цих мов перетинаються. Тому, для більш аргументованого вибору, було проаналізовано була розроблена порівняльна таблиця 3.1.

Таблиця 3.1 - Python, Ruby та JavaScript: порівняння

Особливість	Python	Ruby	JavaScript
Вартість	Безкоштовний і з відкритим вихідним кодом	Безкоштовна та з відкритим вихідним кодом, але потенційно вища вартість розгортання та хостингу через меншу доступність хостингових рішень	Безкоштовний та з відкритим вихідним кодом, але може вимагати додаткових витрат на хостинг та обслуговування сервера Node.js

Продовження таблиці 3.1 - Python, Ruby та JavaScript: порівняння

Читабельність	Легко читається завдяки простому та зрозумілому синтаксису	Читабельна, але використовує більше синтаксичного цукру, що може заплутати початківців	Може бути менш читабельним через асинхронні шаблони програмування та динамічну типізацію
Універсальність	Широкий спектр застосування, особливо гарний для роботи з даними	Сильний у веб-додатках і додатках на стороні сервера	Сильний у веб-додатках
Бібліотеки	Відкриті бібліотеки та фреймворки для вирішення більшості задач	В основному веб-орієнтовані бібліотеки	Величезний набір фреймворків та інструментів

Закінчення таблиці 3.1 - Python, Ruby та JavaScript: порівняння

Підтримка спільноти	Велика та активна спільнота	Активна але менш розвинена спільнота	Одна з найбільших спільнот розробників
Платформа	Крос-платформна сумісність	Крос-платформна сумісність	Для розробки на стороні сервера потрібне середовище Node.js, що додає додатковий рівень складності

Виходячи з аналізу основних характеристик розглянутих мов програмування, Python виділяється як найвигідніший вибір для розробки бота Telegram. Його читабельність, низька вартість і велика кількість готових рішень роблять його ідеальним кандидатом для цієї мети.

3.2.2 Аналіз фреймворків

В Python існує кілька популярних фреймворків за допомогою яких можна розробляти Telegram ботів. Було проведено аналіз їх основних характеристик та виокремлені їх основні недоліки, результати занесені до таблиці 3.2.

Таблиця 3.2 порівняння фреймворків для розробки Telegram ботів

Назва	Характеристики	Недоліки
Aiogram	Асинхронний, підтримує багатопоточність, проміжне програмне забезпечення та FSM для складних сценаріїв користувача, розширені фільтри повідомлень, потужна підтримка спільноти [45].	Більше підходить для досвідчених розробників, які бажають створити складних ботів Telegram із розширеною взаємодією з користувачем.
Botogram	Простота використання, швидке створення простих ботів, оновлений з урахуванням останніх змін в API Telegram Bot [46].	Можливо, це не найкращий вибір для масштабних проектів.
Python Telegram Bot	Простота використання, швидке створення простих ботів, підтримка простих функцій ботів, як-от надсилання повідомлень, обробка введених користувачем даних, керування вбудованими запитами [47].	Відсутність асинхронності.
Pyrogram	Потужний асинхронний фреймворк, обробка сеансів, обробка помилок і автоматичне повторне підключення, розумна система плагінів[45].	Має найвищий поріг входження для розробників та використовується для більш складних проектів.

Проаналізувавши особливості наявних Python фреймворків, для розробки Telegram боту було обрано Aiogram. Не дивлячись на більш складну розробку, факт підтримки складних сценаріїв користувача, можливості гнучких налаштувань робить його фаворитом для розробки високопродуктивної логіки ПЗ. Підтримка асинхронності дає можливість розробити більш ефективну логіку для обробки великої кількості користувацьких запитів. Гарна документованість та наявність великої спільноти розробників зробить процес програмування більш простим і зрозумілим.

3.2.3 Вибір хостингу

Обираючи хостинг для оренди серверу, особливу увагу було звернуто на популярність компанії на ринку, наявність функцій для зручного системного адміністрування, підтримку, безпеку даних, якість та ціну послуг.

Hetzner Hosting - провідний оператор веб-хостингів та обробки даних в Європі. Має дата центри в Німеччині, Фінляндії та США. Серед системних адміністраторів він відомий своїми надійними та економічно ефективними рішеннями. Hetzner надає широкий спектр послуг, що задовольняють різноманітні потреби, від окремих проектів до масштабних корпоративних рішень [48].

Види веб хостингу що підтримуються

Спільний веб-хостинг (Shared): ідеально підходить для невеликих і середніх веб-сайтів, додатків або скриптів, пропонуючи спільне середовище, де кілька веб-сайтів знаходяться на одному сервері. Це економічно вигідно та зручно для користувача, часто постачається з панеллю керування, як-от cPanel [49].

Хостинг VPS (виртуальний приватний сервер): надає віртуальне середовище з виділеними ресурсами. Він пропонує більше можливостей

керування та налаштування, ніж спільний хостинг, підходить для веб-сайтів із середнім трафіком [49].

Виділені сервери (VDS): пропонує цілі фізичні сервери, призначені для одного клієнта. Ця послуга ідеально підходить для веб-сайтів і програм із високим трафіком, які вимагають повного контролю над серверним середовищем, однак таке ришення буде най витратнішим [49].

Хмарний хостинг (Cloud): надає гнучкі рішення для хостингу, які дозволяють масштабувати ресурси за вимогою. Він підходить для додатків із коливанням трафіку та для користувачів, які віддають перевагу моделі ціноутворення з оплатою за використання [49]. В розробленому ПЗ буде використано саме цей тип послуг.

Послуги спільного розміщення: в клієнтів є можливість розмістити власні сервери в центрах обробки даних Hetzner, отримуючи вигоду від їхньої інфраструктури та підключення та зберігаючи контроль над своїм обладнанням [49].

Hetzner пропонує різноманітні операційні системи для своїх послуг хостингу, включаючи Ubuntu, Debian, Fedora, CentOS, Rocky Linux і AlmaLinux [50].

Підтримка клієнтів хостингу здійснюється на гідному рівні. Залежно від оплаченого пакету послуг, пропонуються різні канали зв’язку, такі як електронна пошта, телефон та подача заявок в службу підтримки.

Важливим є те, що за замовчуванням Hetzner надає можливість резервного копіювання даних в ручному режимі або налаштувавши розклад.

Hetzner безкоштовно надає захист безпеки від DDoS-атак, та дбає про захист безпеки своїх серверів як фізично так і за допомогою апаратного

забезпечення [49]. Також, маючи належність до Європейської юрисдикції підтримує закони і правила ЄС про збереження і обробку даних.

3.2.4 Середа розробки

PyCharm - це комплексна IDE (інтегрована середа розробки), яка пропонує розширені можливості для розробки на Python.

Перелік основних особливостей IDE наведено нище.

Розумний редактор коду: включає в себе перевірку синтаксису коду, підсвічування помилок та швидкі виправлення, а також автоматичний рефакторинг коду та широкі можливості навігації. В останніх версіях додано підтримку помічника з штучним інтелектом (ШІ)[51].

Підтримка мов: Крім Python підтримує розробку також на JavaScript, CoffeeScript, TypeScript, CSS, популярних мовах шаблонів тощо [51].

Спеціальна підтримка фреймворків: PyCharm пропонує чудову підтримку фреймворків для сучасних фреймворків веб-розробки, таких як Django, Flask, Google App Engine, Pyramid і web2py, також має підтримку Django, інструментів manage.py і appcfg.py [51]

Інтеграційні інструменти розробника: PyCharm інтегрується з такими системами відстеження помилок, як Atlassian JIRA, JetBrains YouTrack, Lighthouse, Pivotal Tracker, GitHub, Redmine і Trac. Він також дозволяє легко налаштувати необхідне середовище Python для проектів, підтримує інтерактивні консолі Python та інтегрується з популярними платформами тестування [52].

Підтримка веб-розробки: забезпечує широку підтримку повноцінної розробки за допомогою Vagrant, SSH і Docker, а також використання WebStorm для допомоги в інтелектуальному кодуванні. Для JavaScript і TypeScript, має вбудований відладчик коду на стороні клієнта [53].

3.3 Серверна частина програмного забезпечення

Під час розробки серверної частини ПЗ була використана мова програмування Python. Реалізація Telegram боту відбувалася на основі асинхронного фреймворку Aiogram.

3.3.1 Реєстрація Telegram боту

Для того щоб створити будь якого Telegram-бота, його спершу потрібно зареєструвати. Робиться це за допомогою головного чат-боту Telegram BotFather (рисунок 3.1). Це офіційний сервіс Telegram, і управління всіма створеними ботами ведеться тільки через нього.

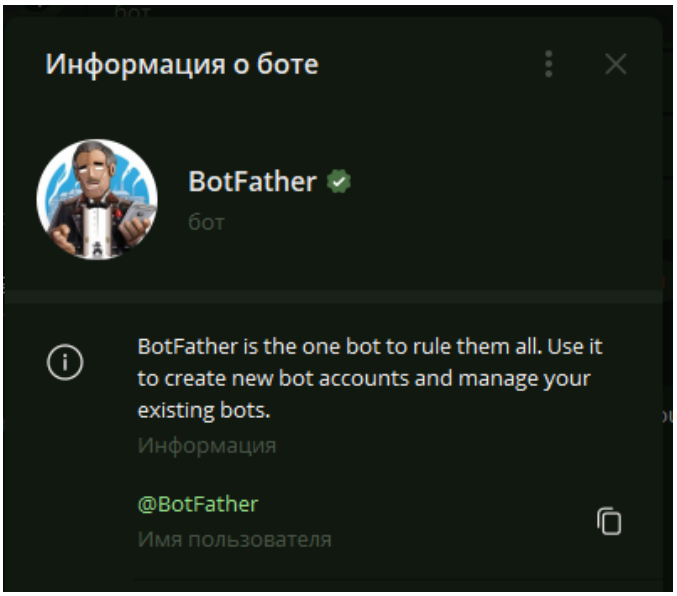

Рисунок 3.1 – Інформація про BotFather

Потрібно розпочати з ним діалог та за допомогою команд пройти процедуру реєстрації. На кожному наступному кроці BotFather буде показувати, які дії потрібно робити далі. Потрібно буде вигадати ім'я та нікнейм, що має кінчатися на «bot». Якщо бажаний нікнейм вже був чимось зайнятий буде відповідне повідомлення. В якості нікнейму було обрано «@better_tg_with_stegobot». Далі в розділі «Про нас» було додано опис боту та додана фотографія профілю. На наступному кроці для створеного боту було

отримане HTTP API. За допомогою нього ведеться все управління ботом, тому його не можна показувати стороннім.

3.3.2 Розробка Python проекту

Після реєстрації бота та отримання токену управління була почата його розробка та тестування.

В середовищі розробки Pycharm було створено новий проект, та віртуальне середовище. Локальним інтерпретатором було обрано мову програмування Python 3.11. Активувавши віртуальне середовище командою «python –m venv venv», далі за допомогою команди «pip install (назва бібліотеки)» було встановлено всі необхідні бібліотеки.

Була створена необхідна ієрархія (файлова структура) директорій та основні файли «create_bot.py» та «main.py».

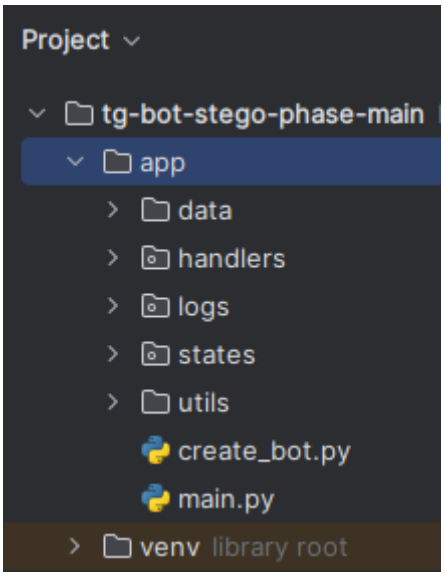

Рисунок 3.2 - Файлова структура Telegram боту

Написання коду боту було розпочате з файлу «main.py». Спочатку було імпортовано модуль «executor» з бібліотеки «aiogram», та (для зручності читання коду) модулі з інших файлів.

```
from aiogram import executor

from app.handlers import dp
from app.logs import logging
```

Рисунок 3.3 - Імпорт модулів

Далі була використана конструкція, метою якої є не допустити запуску цього скрипту з іншого файлу. Це потрібно щоб не зруйнувати логіку роботи програми.

```
if __name__ == '__main__':
    logging.info("Starting bot")
    executor.start_polling(dp, skip_updates=True)
    logging.info("Stopping bot")
```

Рисунок 3.4 - Обмеження запуску скрипту з інших файлів

Проаналізувавши файлову структуру розробленого - боту можна зрозуміти логіку його роботи. Умовно алгоритм роботи представляє собою такий порядок:

Після запуску боту через команду «python main.py», головний файл «main.py» викликає запуск обробника повідомлень «dp», який постійно запитує сервера Telegram API нових повідомлень. Параметр «skip_updates=True» дозволяє відключити виконання команд запущених користувачами в час коли бот не працював. Також, одразу після запуску боту, в консоль адміністратору відправляється повідомлення що бот запущений. Це реалізовано за допомогою Python бібліотеки loguru.

3.3.3 Реалізація клієнтської частини боту

Користувачі запускають бота за допомогою команди «/start» або кнопки запустити. Для того щоб забезпечити зручне користування ботом, було вирішено розробити клавіатуру. Для реалізації цієї цілі буде розроблене меню (з вибором функцій боту).

Після запуску боту, користувач бачить меню вибору функцій та початкову клавіатуру, що складається з двох кнопок:

- Приховати дані в аудіо;
- Вилучити дані.

Клавіатура реалізована за допомогою обробників подій. Вони в свою чергу реалізовані в виді асинхронних функцій обернених в декоратори. Згідно з документацією Aiogram кнопки реалізовані у вигляді списку об’єктів. Кожна кнопка має текстовий атрибут. Для більшої зручності, клавіатура вирівнюється під розмір екрану. Також кожний логічний пункт інтерфейсу несе свій поточний стан.

Приклад реалізації клавіатури одного з меню чат-боту зображено на рисунку 3.5.

```
1 usage
@dp.message_handler(commands=['start'])
async def start_cmd(message: types.Message):
    buttons = [
        KeyboardButton(text="Приховати дані в аудіо"),
        KeyboardButton(text="Вилучити дані"),
    ]

    reply_markup = ReplyKeyboardMarkup(keyboard=[buttons],
                                       resize_keyboard=True)

    await UserStates.main_menu.set()
    await message.answer( text: "<b> Оберіть функцію</b>",
                         parse_mode=types.ParseMode.HTML,
                         reply_markup=reply_markup)


@dp.message_handler(Text(equals=["← Повернутися в початкове меню"]), state='*')
async def go_menu(message: types.Message, state: FSMContext):
    await UserStates.main_menu.set()
    await start_cmd(message)
```

Рисунок 3.5 - Код реалізованої клавіатури основного меню

В результаті клавіатура мала зовнішній вигляд, зображений на рисунку

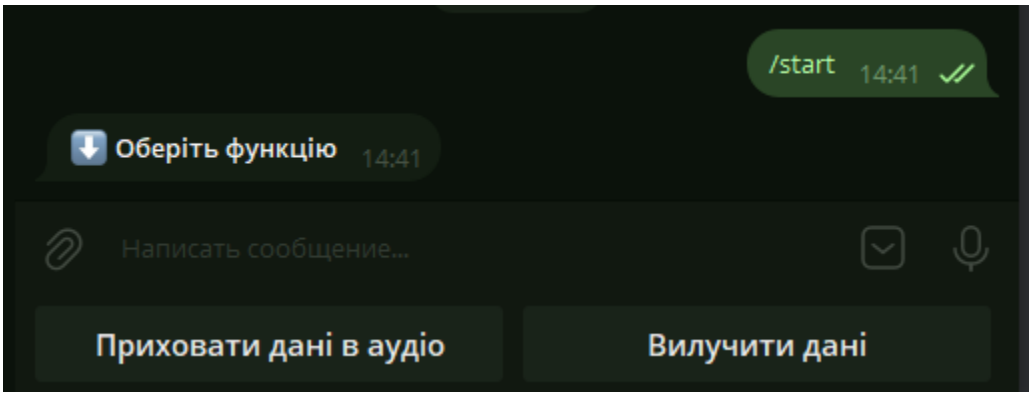

Рисунок 3.6 - Початкове меню боту

Наступним кроком було розроблено обробник для основних функцій боту, вбудовування та вилучення інформації з аудіо файлу.

Код для цього було реалізовано за тією ж логікою, що й для початкового меню, фрагмент коду представлено на рисунку 3.7. Щоб попередити появу неочікуваних порушень роботи системи через помилки з боку користувача, були додані логічні перевірки типу надісланого файлу за допомогою конструкціїх “if/else”.

```
@dp.message_handler(Text(equals=["Вилучити дані"]), state=UserStates.main_menu)
async def start_decrypt(message: types.Message, state: FSMContext):
    await UserStates.send_decrypt_file.set()
    await state.update_data(file_path="")
    await message.answer("Відправте WAV файл, з якого хочете вилучити дані")


@dp.message_handler(state=UserStates.send_decrypt_file, content_types=ContentType.DOCUMENT)
async def handle_document_decrypt(message: types.Message, state: FSMContext):
    user_id = message.from_user.id
    if message.document.mime_type == 'audio/wav' or message.document.file_name.endswith('.wav'):
        file_path_decrypt = await save_file_decrypt(message.document.file_id)
        await state.update_data(file_path_decrypt=file_path_decrypt)
        print(f"save_file decrypt {file_path_decrypt}")
        await UserStates.send_text_decrypt.set()
        await message.answer("Файл принятий. Тепер відправте segment_width для розшифрування.")
    else:
        buttons = [
            KeyboardButton(text="← Повернутися в початкове меню")
        ]

        reply_markup = ReplyKeyboardMarkup(keyboard=[buttons],
                                           resize_keyboard=True)
        await message.answer( text: "Будь ласка, відправте файл в форматі WAV.",
                             reply_markup=reply_markup)
```

Рисунок 3.7 - фрагмент коду меню зашифрування даних

Також реалізовані кнопка для можливості повернутися в початкове меню після закінчення виконання необхідної операції, її зовнішній вигляд зображено на рисунку 3.8.

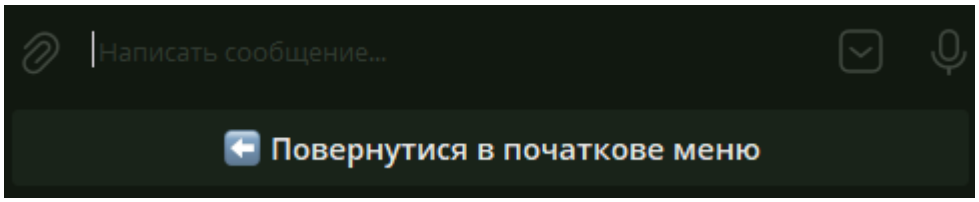

Рисунок 3.8 - кнопка “Повернутися в початкове меню”

Таким чином була реалізована клієнтська частина чат боту для зашифрування та розшифрування даних.

3.3.4 Реалізація серверної частини боту

Основні задачі які повинна вирішувати серверна частина Telegram боту:

- Отримувати та розпізнавати команди користувача;
- Обробляти команди користувача;
- Вести лог.

Для того щоб краще керувати потоком взаємодії користувача з ботом в програмний код боту було побудовано з урахуванням концепції кінцевого автомату (Finite-state machine). З допомогою цього буде досягнуто:

- Кращого структурування діалогів;
- Уникнення плутанини в роботі з обробкою повідомлень;
- Простота розробки та відладки;
- Краща гнучкість та масштабованість.

В цьому контексті було реалізовано модуль що відповідає за стан користувача, фрагмент коду зображено на рисунку 3.9.

```
class UserStates(StatesGroup):
    main_menu = State()
    send_encrypt_file = State()
    send_text_encoding = State()
    send_encrypt_file = State()
    send_decrypt_file = State()
    send_text_decrypt = State()
```

Рисунок 3.9 - Реалізація класу стану користувача

В першу чергу була реалізована функція приховування даних в аудіофайл. Для того щоб приховати дані, користувач повинен натиснути кнопку “Приховати дані в аудіо”. Наступним повідомленням він повинен надіслати необхідний аудіо файл формату WAV. потім написати текст для приховування. На це бот відповість повідомленням з параметром “segment_width”, який необхідний для розшифрування повідомлення та файлом з прихованими даними. Результат роботи зображений на рисунку 3.10.

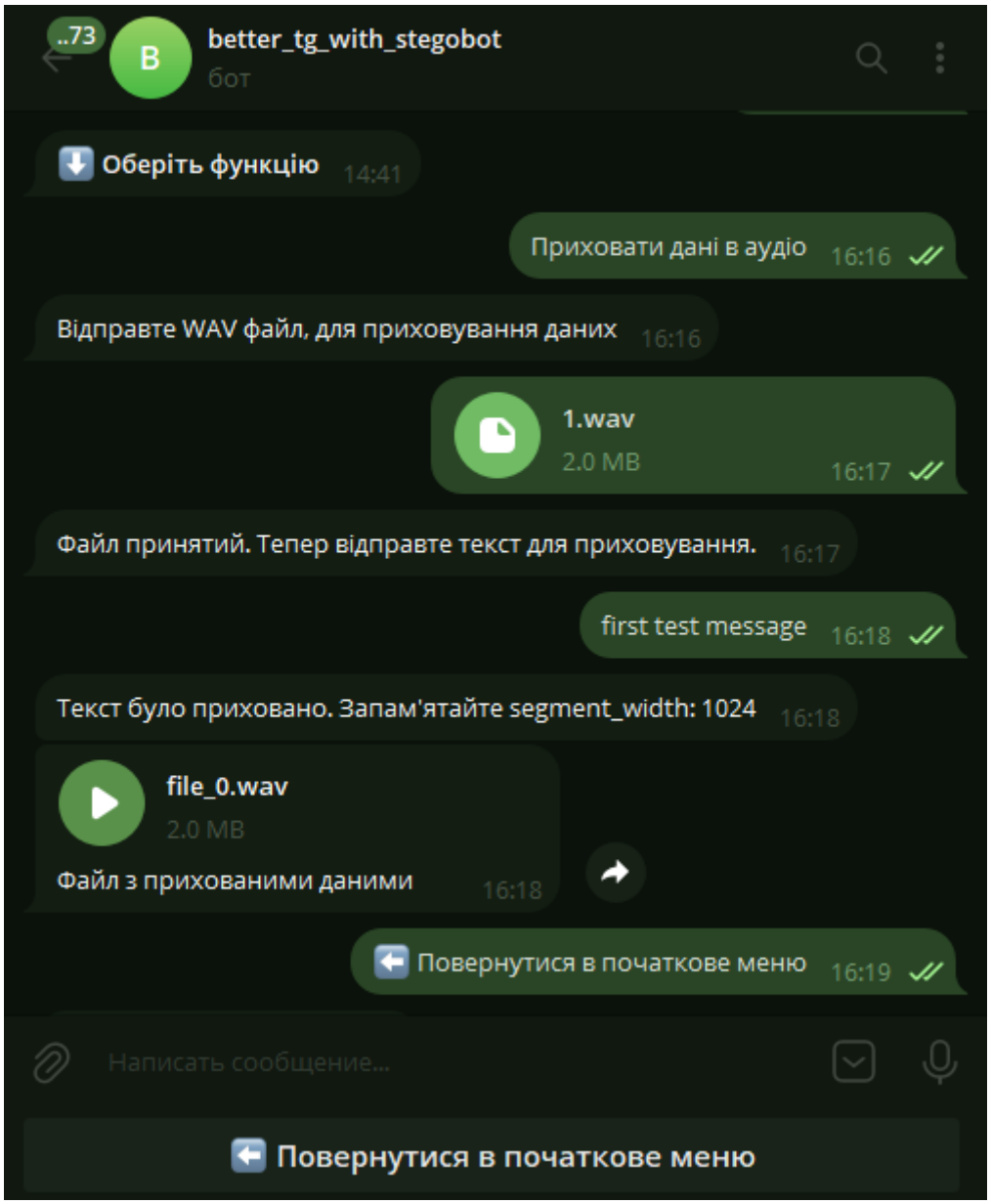

Рисунок 3.10 - Результат виконання функції приховування даних

Як видно на рисунку 3.10. розмір файла до вбудовування в нього повідомлення дорівнює розміру файлу після перетворень, це може говорити про якісну роботу стеганографічного перетворення.

Наступним було розроблено функцію вилучення даних. Для того щоб нею скористатися потрібно обрати відповідний пункт початкового меню, відправити аудіо файл формату WAV з вже вбудованими даними, наступним повідомленням надіслати значання параметру “segment_width”. Наступним

повідомленням бот відправить вилучине повідомлення. Результат роботи графічно зображено на рисунку 3.11.

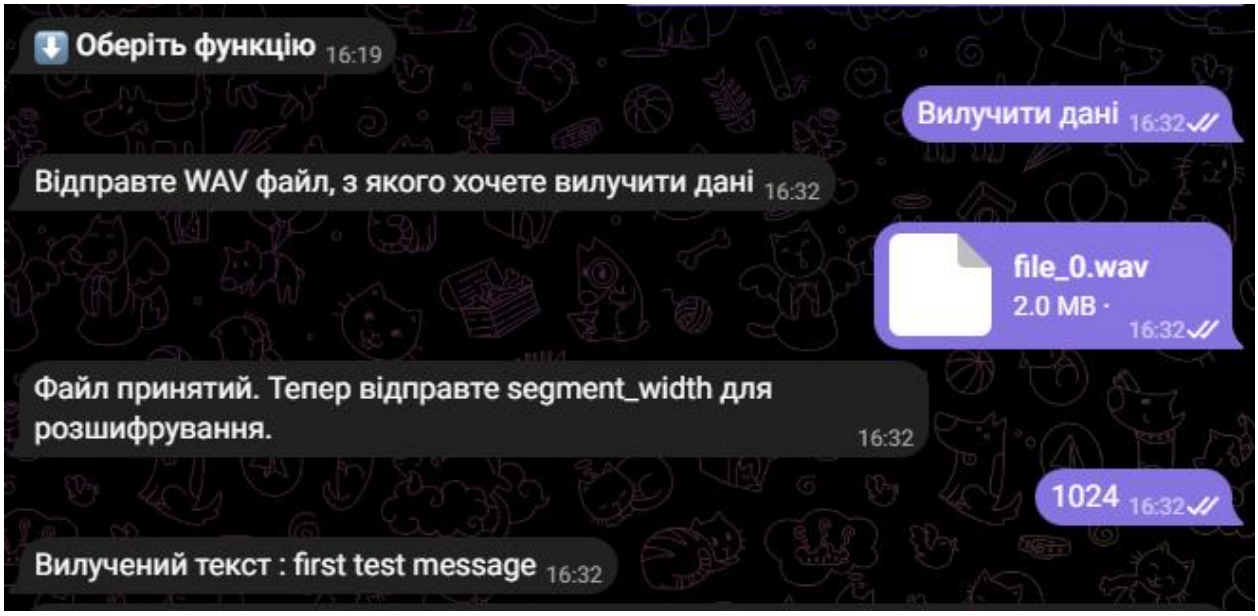

Рисунок 3.11 - Результат роботи функції вилучення.

Логування було реалізовано в виді окремого модуля з використанням бібліотеки loguru. Логи відображаються в консолі адміністратора та записуються в спеціальний файл log.log. Фрагмент коду зображено на рисунку 3.12.

```
logging.add("logs/logs.log",
            rotation="500 MB",
            backtrace=True,
            diagnose=True,
            format="<green>{time:YYYY-MM-DD at HH:mm:ss}</green> | <level>{level}</level> | <level>{function}</level> : <level>{message}</level>",
            colorize=False  # Включаем цветовую разметку
            )
logging.opt(colors=True)
```

Рисунок 3.12 - - Реалізація логування.

Таким чином було реалізовано наступні функції Telegram боту

- Вбудовування даних в аудіофайл;
- Вилучення даних.

3.3.4.1 Реалізація стеганографічного методу фазового кодування

Для вичислення математичних операцій було використано бібліотеки math, cmath numpy, для роботи з двійковими даними struct, а для роботи з файлами типу wave- однойменну бібліотеку.

Реалізація алгоритму вбудовування даних

Вбудовування даних в аудіофайл формату WAV було реалізоване в виді функції “hide” логіку та послідовність її роботи можна описати наступним чином:

- Файл завантажується, і з нього вилучаються аудіодані (лівий канал використовується для налаштування повідомлень).
- Визначається ширина сегмента на основі довжини повідомлення.
- Аудіосигнал ділиться на сегменти.
- На кожному сегменті виконується перетворення Фурье (ДПФ) для перекладу сигналу в частотну область, вилучаються амплітуди і фази.
- У фазі кожного сегмента вбудовується повідомлення (модифікація фази здійснюється на основі битів повідомлення).
- Сегменти зі зміненими фазами об'єднуються в аудіосигнал і зберігаються в новому файлі.

Реалізація алгоритму вилучення даних

Для реалізації алгоритму вилучення даних з аудіофайлу була створена функція “recover”, вона реалізує таку послідовність:

- Файл із вбудованим повідомленням завантажується.
- Аудіосигнал ділиться на сегменти з відомою шириною.
- Замінюється перетворення Фур'є до першого сегменту.
- На основі змін фаза витягується приховане повідомлення.
- Последовательность битов преобразуется обратно в текст.

Таким чином, під час розробки ПЗ, що являє собою Telegram бот, були сформульовані необхідні вимоги. Аргументовано обрання конкретних технологій розробки, проведена розробка серверної частини ПЗ

Таким чином, з метою підвищення захищеності месенджеру Telegram , було розроблено програмне забезпечення у вигляді чат-боту. Він реалізує функції вбудовування та вилучення даних з аудіо файлу формату WAV, за допомогою стеганографічного методу фазового кодування. Розроблений бот

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО TELEGRAM-БОТА

Перед тим як масово використовувати будь який підхід в реальному житті потрібно не тільки протестувати його на наборі справжніх прикладів, а й дослідити та розуміти його робочі показники. Цифрова аудіо стеганографія, а також її стегоаналіз досі не має широкого розповсюдження. Якщо, для LSB відповідне ПЗ ще можна знайти, то для більш складного методу фазового кодування - готові якісні програмні реалізацій для комплексного тестування відсутні.

4.1 Розробка методики експерименту

Для реалізації тестування було обрано мову програмування Python.

Перед дослідженням якості роботи розробленої системи, потрібно впевнитися в тому, що взагалі система успішно працює. Тому першим кроком дослідження розробленої системи буде перевірка надійності її роботи. Далі будуть проведені тести пікового співвідношення сигналу до шуму (Peak Signal to Noise Ratio - PSNR) та досліджений коєфіцієнт бітових помилок (Bit Error Rate - BER). Також буде проведено аналіз отриманих результатів для того щоб дослідити розроблений Telegram бот, та метод підвищення захищеності Telegram який за допомогою нього реалізується.

В таблиці 4.1 наведено результат перевірки успішності вбудовування та вилучення повідомлень в аудіо файлів різної довжини. Дослідження було проведено для 10 аудіо файлів формату WAV різної довжини. В кожен аудіо файл було проведене вбудування тексту розміром 30 Kb. Текст містить літери цифри та пробіли.

Проаналізувавши результати проведеного дослідження, можна сказати, що всі вбудовування та вилучення повідомлень були успішними. Це свідчить про якість роботи розробленої стеганосистеми.

Таблиця 4.1 Результат успішності вбудовування та вилучення повідомлень

№	Розмір (Kb)	Вбудування	Вилучення
1	123,6	Успішно	Успішно
2	248,4	Успішно	Успішно
3	486	Успішно	Успішно
4	723,6	Успішно	Успішно
5	972	Успішно	Успішно
6	1240,8	Успішно	Успішно
7	1808,4	Успішно	Успішно
8	2428	Успішно	Успішно
9	6201,6	Успішно	Успішно
10	7338	Успішно	Успішно

4.1.1 Реалізація PSNR тестування

Пікове співвідношення сигналу до шуму (PSNR) – кількісно визначає співвідношення між максимально можливою потужністю сигналу та потужністю спотворюючого шуму, який впливає на точність його представлення [30]. Зазвичай, цей показник використовується для оцінки якості зображень, які були піддані стисненням з втратами, а потім відновлені [31]. У контексті аналізу стеганографії аудіофайлів, PSNR можна використовувати для оцінки якості передачі та прийому аудіоданих, надаючи уявлення про якість алгоритму вбудовування даних. Ця характеристика може бути важливою для дослідження роботи стеганосистеми, в процесі розуміння непомітності вбудовування повідомлення для стегоконтейнерів у вигляді аудіо файлів. PSNR виражається в децибелах (дБ) і це можна вважати показником якості сигналу після внесених змін. За допомогою цього тесту

можна буде прослідкувати якість звуку після вбудовування повідомлення. Виходячи з досліджень [39], чим менший обсяг даних вбудовується в файл, в порівнянні з розміром файла контейнера, тим більший PSNR буде сгенерований стеганографічною системою. Тобто показник можливого створеного шуму (змін початкового файлу) буде менший.

Для визначення PSNR (у дБ) можна скористатися середньо квадратичною помилкою (Mean Squared Error - MSE). Враховуючи безшумний сигнал і його шумову апроксимацію, MSE обчислюється як середній квадрат різниці між вихідним і зашумлениним сигналом. PSNR визначається як логарифмічна величина за шкалою децибел. Щоб оцінити PSNR аудіофайлу, необхідно порівняти вихідний сигнал з його погіршеною версією, в нашому випадку це буде аудіофайл з вже вбудованим повідомленням. Розрахунок PSNR виконується за математичною моделлю представленою у формулі 1:

$$\text{PSNR} = 20 \times \log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right) \quad (1)$$

Де, MAX_I – максимально можливий рівень інтенсивності (для аудіо це відповідає максимально можлиивій амплітуді звукового сигналу); MSE - це середня квадратична помилка, розрахована як середня квадратична різниця між початковим та погіршеним квадратичним сигналом.

Для обчислення PSNR показнику був реалізований скрипт на мові програмування Python. Код скрипту представлено в додатку А.

В проведеному дослідженні було використано 10 аудіо файлів формату WAV різного розміру. Для всіх 10 зразків аудіофайлів було взято однаковий розмір вбудованого повідомлення та однакову ширину сегменту що використовувався. Такі характеристики взятої вибірки дозволяють прослідкувати вплив сегментації на аудіо стего контейнер. Також розташувавши отримані результати для кожного аудіо файлу у порядку зростання, можна прослідкувати, який є взаємозв’язок з показником PSNR. Розмір повідомлення дорівнює 30 Kb. Розмір сегменту 96 Kb.

Результати проведеного PSNR тестування для аудіофайлів різних розмірів наведені в таблиці 4.2.

Таблиця 4.2 Результат PSNR тестування

№	Розмір контейнеру (Kb)	Розмір сегменту (Kb)	Розмір повідомлення(Kb)	PSNR (dB)
1.	123,6	96	30	62,53
2.	248,4	96	30	57,67
3.	486	96	30	61,12
4.	723,6	96	30	59,43
5.	972	96	30	56,52
6.	1240,8	96	30	57,93
7.	1808,4	96	30	59,96
8.	2428	96	30	57,32
9.	6201,6	96	30	53,22
10.	7338	96	30	41,28

Аналізуючи отримані результати можна побачити, що для 10 протестованих аудіофайлів, найбільше значення PSNR 62,53 dB, найменше 41,28 dB. З цього можна зробити висновки, що розроблена стеганосистема показує гарний результат вбудовування повідомлень. Значення PSNR є досить великим що говорить про якість утвореного стего контейнера, та незначні зміни. Цей факт дає можливість припускати, що утворені стегано контейнери будуть менш помітними для зловмисників.

Мета наступного тестування в тому щоб дослідити можливі допустимі співвідношення при яких будуть зберігатися прийнятні значення PSNR. Для цього потрібно прослідкувати залежність значення PSNR від зміни (поступового збільшення) розміру вбудованого повідомлення відносно розміру аудіо файлу. Для проведеного експерименту використані десять аудіо файлів формату WAV. Тестування проводиться в 10 етапів для кожного з 10-ти аудіофайлів. Під час кожного наступного етапу розмір вбудованого повідомлення збільшується на 5 відсотків (від 5 до 50 %).

Для вибірки з 10 аудіо контейнерів виповнено тестування показнику PSNR з поступовим збільшенням розміру (на 5% на кожному з 10 етапів) повідомлення яке вбудовується. Для кращого розуміння кореляції PSNR з іншими показниками буде проаналізовано та внесено до таблиць результат тестування для 2 різних аудіо контейнерів.

Таблиця 4.3 Результат тестування для 1-го аудіо контейнеру

№	Розмір контейнеру (B)	Розмір повідомлення (%)	Розмір повідомлення (B)	PSNR (dB)
1.	291060	5	14553	72,97
2.	291060	10	29106	69,52
3.	291060	15	43659	65,08
4.	291060	20	58212	61,52
5.	291060	25	72765	62,12
6.	291060	30	87318	61,48
7.	291060	35	101871	59,65
8.	291060	40	116424	51,22
9.	291060	45	130977	47,04
10.	291060	50	145530	33,12

Таблиця 4.4 Результат тестування для 10-го аудіо контейнеру

№	Розмір контейнеру (B)	Розмір повідомлення (%)	Розмір повідомлення (B)	PSNR (dB)
1.	166855630	5	8342781,5	112,36
2.	166855630	10	16685563	108,05
3.	166855630	15	25028344,5	63,11
4.	166855630	20	33371126	55,07
5.	166855630	25	41713907,5	48,09
6.	166855630	30	50056689	42,98
7.	166855630	35	58399470,5	42,13
8.	166855630	40	66742252	40,02
9.	166855630	45	75085033,5	38,49
10.	166855630	50	83427815	23,87

Результати тестування для всіх 10 аудіофайлів були представлені в виді графіку на рисунку 4.1.

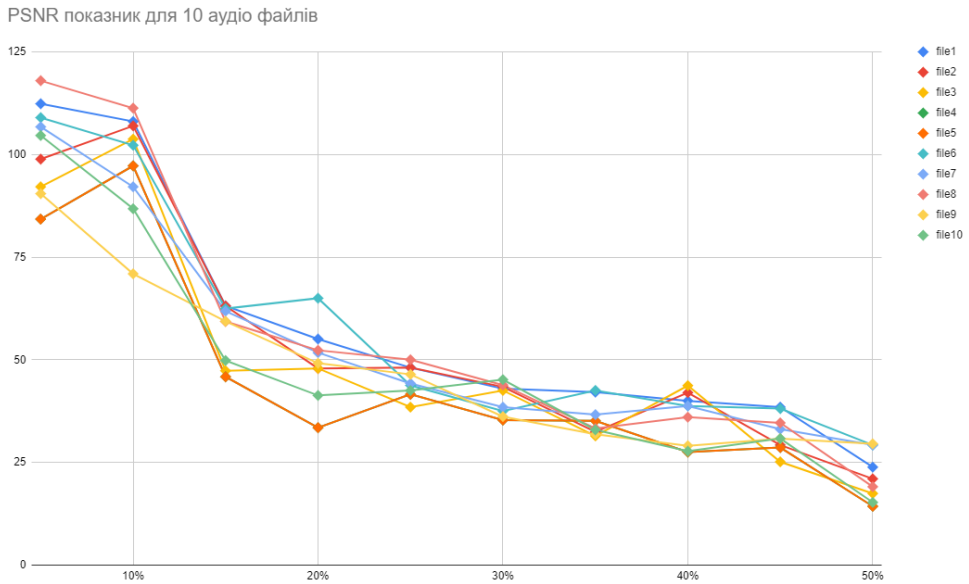

Рисунок 4.1 – PNSR показник для 10-ти тестів 10 аудіофійлів

4.1.2 Реалізація BER тестування

Коефіцієнт бітових помилок або ще частота бітових помилок (Bit Error Rate – BER) – це відношення кількості бітових помилок на одиницю часу в цифровій системі передачі. За допомогою нього можна кількісно виміряти точність і надійність передачі даних, а за допомогою цього, забезпечити розуміння якості каналу зв'язку [54]. У контексті нашої роботи за допомогою BER тестування можна оцінити якість реалізації алгоритму вбудовування та вилучення повідомлення із стего контейнеру. Тобто можна зрозуміти чи відбувається спотворення інформації, яка вбудовувалася, при вилученні. В разі, коли помилки повністю відсутні значення показника BER буде рівне нулю, це вважається найкращим результатом. BER - тест для аудіо файлів має важливе значення для перевірки забезпечення точності та надійності передачі цифрового аудіо. Особливо для стеганографічної системи, де в якості контейнеру використовується файл формату WAV. Втрата даних або стиснення такого файлу неминуче призведуть до втрати якості звуку та, що найголовніше, впливу на вбудоване повідомлення.

Для вирахування BER потрібно почергово порівняти кожен біт початкового аудіо файлу з аудіо файлом у який вже вбудоване повідомлення, щоб визначити, чи є різні (помилкові) біти чи ні. BER обчислюється шляхом ділення кількості бітових помилок на загальну кількість переданих бітів протягом досліджуваного інтервалу часу [55]. Зазвичай він виражається як співвідношення або відсоток, що представляє ймовірність бітових помилок, що виникають під час передачі [54]. Для дослідження частоти бітових помилок було обрано середу MATLAB та реалізацію BER тесту за допомогою додатку [56].

BER тестування було проведено для повідомлення, яке вбудовувалося та вилучалося з стегоконтейнеру. Розмір повідомлення 30 кілобайт (Kb). Вимірювання проводилися для різних контейнерів протягом 10 разів. В таблиці 4.5 представлено розмір повідомлення, що вбудовувалося та що

вилучалося з стегоконтейнеру та величину отриманого значення BER. Можна бачити, що всі показники BER тесту дорівнюють нулю, це свідчить про те що помилки при перетвореннях (вбудуванні і вилученні) відсутні, тобто про гарну роботу розробленої стеганосистеми.

Таблиця 4.5 Результати BER - тесту

№	Розмір вбудованого повідомлення (Kb)	Розмір вилученого повідомлення (Kb)	BER
1.	30	30	0
2.	30	30	0
3.	30	30	0
4.	30	30	0
5.	30	30	0
6.	30	30	0
7.	30	30	0
8.	30	30	0
9.	30	30	0
10.	30	30	0

4.2 Оцінка набору тестових даних

Для оцінки роботи програмної реалізації загалом було протестовано 20 аудіо записів формату WAV. Обрані аудіофайли мали різний розмір (від 123,6 Kb до 166,85 Mb). Для вбудовування використовується повідомлення різного розміру (від 30 Kb до 83,42 Mb). Таким чином, вибірка сформована досить обширно, це повинно дати можливість отримати максимально повні результати досліджень.

4.3 Аналіз результатів експерименту

Аналізуючи показники результатів всієї вибірки дослідженнь можна зробити висновки:

1) Розроблена стегосистема має показник успішності вбудовування та вилучення інформації (З 10 вбудувань всі 10 були успішними), також про це свідчить отриманий коефіцієнт бітових помилок;
2) Тестування пікового співвідношення сигналу до шуму (PSNR) в загальному має високий результат – це каже про гарну якість отриманих стегано контейнерів, а отже успішність стеганографічних перетворень;
3) Проаналізувавши показники PSNR для різних співвідношень розміру стеганоконтейнеру і вбудованого повідомлення – прийнятні значення PSNR зберігаються коли розмір повідомлення до 45% від обсягу файлу;
4) Результати BER тесту показали, що помилки при перетвореннях (вбудуванні і вилученні) відсутні, це свідчить про гарну роботу розробленої стеганосистеми;
5) Значення PSNR зростає разом зі зменьшенням розміру вбудованого повідомлення.

ВИСНОВКИ

Проведений в роботі аналіз захищеності месенджеру Telegram показав, що наявні вразивості потребують застосування методів підвищення захищеності на основі застосування стеганографічних алгоритмів.

В межах даної кваліфікаційної роботи магістра були виконані поставлені задачі. По-перше, була досліджена захищеність Telegram месенджеру, визначені основні недоліки та запропоновано підвищення рівня захисту інформації за допомогою стеганографічних алгоритмів. Проведено порівняльний аналіз стеганографічних методів для досягнення поставленої на дослідження задачі. В результаті було обрано стеганографічний метод фазового кодування. Також було проаналізовано формати аудіофайлів, та обгрунтовано вибір в якості стеганоконтейнеру саме аудіофайлів формату WAV. Також було обгрунтовано створення з метою зручності для користувачів програмного забезпечення в вигляді саме Telegram боту.

По-друге, було розроблено концептуальну модель програмного забезпечення та сформульовано основні вимоги до нього. Було обрано клієнт-серверну архітектуру в якості патерну системної архітектури, реалізовану за допомогою компонентного стилю системної архітектури.

По-третє, в процесі розробки Telegram боту, до нього були сформульовані та виконані такі основні вимоги, як швидкість відповіді, здатність до масштабування, надійність та здатність обробки великих об'ємів даних. Було проаналізовано та обгрунтовано доцільний вибір мови програмування Python в якості платформи для розробки. Також, було реалізовано серверну та клієнтську частини боту. Для клієнтської частини було розроблено користувацький інтерфейс та, за допомогою механізму FSM, логіка меню управління ботом. На серверній частині боту було реалізовано

логіку та механізм виконання функцій вбудовування та вилучення даних з аудіо файлів формату WAV.

В-четверте, було експериментально досліджено ефективність розробленого Telegram боту. Для цього використовувались PSNR та BER тести. Вибірка даних для тесту складалась, загалом, з 20-ти аудіофайлів різної довжини (від 123,6 Kb до 166,85 Mb) та повідомлень різного розміру (від 30 Kb до 83,42 Mb). Аналіз результатів машинного тестування дають можливість зробити висновок, що розроблена стегосистема здатна працювати без збоїв та забезпечити низьку модифікацію (при розмірі вбудованого повідомлення 5% від всього контейнеру – PSNR показник в середньому 100 dB) стегано контейнеру. Про це свідчить значення протестованого показника PSNR (від 23 %). Також, в результаті тестування, було підтверджено залежність показнику PSNR від обє'му вбудованих даних. Значення PSNR зростає разом зі зменьшенням розміру вбудованого повідомлення.

Таким чином, використання стеганографічних алгоритмів для приховування передачі конфіденційних даних дає змогу підвищити захищеність Telegram месенджеру. В якості напряму наступних досліджень можна вказати аналіз стеганосистем, що використовують поєднання пост-квантових криптографічних перетворень і стеганографічних алгоритмів.

ПЕРЕЛІК ВИКОРСТАНИХ ДЖЕРЕЛ

1. SoK: Secure Messaging / N. Unger et al. 2015 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 17–21 May 2015. 2015. URL: https://doi.org/10.1109/sp.2015.22 (date of access: 29.11.2023).
2. Viber Encryption Overview. Rakuten Viber. URL: https://www.viber.com/app/uploads/viber-encryption-overview.pdf (date of access: 08.09.2023).
3. Facebook. Messenger Secret Conversations. Technical Whitepaper. URL: https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf (date of access: 08.09.2023).
4. Telegram APIs. Telegram. URL: https://core.telegram.org (date of access: 08.09.2023).
5. WhatsApp Encryption Overview. Technical white paper. WhatsApp. URL: https://scontent.fiev21-1.fna.fbcdn.net/v/t39.8562-6/384251896_820338303082371_8514785982310046047_n.pdf?_nc_cat=100&ccb=1-7&_nc_sid=e280be&_nc_ohc=xQXHEXgxGvgAX_IWcVI&_nc_oc=AQmHfAXcJWx3Uk85423KC3YizdEN-rtQ_KUQJThfPY4vCwpNiyDbVA-bwoScJ1wiVTw&_nc_ht=scontent.fiev21-1.fna&oh=00_AfCJBUI4sl0qzdGjT96zLGwwgKqnA-MFCf76XkWrNiAj1w&oe=656D4951 (date of access: 29.11.2023).
6. Політика конфіденційності. Telegram. URL: https://telegram.org/privacy/ua (дата звернення: 10.09.2023).
7. Miculan M., Vitacolonna N. Automated verification of telegram’s mtproto 2.0 in the symbolic model. Computers & security. 2022. P. 103072. URL: https://doi.org/10.1016/j.cose.2022.103072 (date of access: 29.11.2023).

8. Alatawi M., Saxena N. SoK: An Analysis of End-to-End Encryption and Authentication Ceremonies in Secure Messaging Systems. WiSec '23: 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Guildford United Kingdom. New York, NY, USA, 2023. URL: https://doi.org/10.1145/3558482.3581773 (date of access: 29.11.2023).
9. steganography - Glossary | CSRC. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/glossary/term/steganography (date of access: 29.11.2023).
10. Bogos, Corina-Elena & Mocanu, Razvan & Simion, Emil. A security analysis comparison between Signal, WhatsApp and Telegram. 2023. P. 1–15. URL: https://www.researchgate.net/publication/367350335_A_security_analysis_comparison_between_Signal_WhatsApp_and_Telegram (date of access: 10.09.2023).
11. Regular reminder that Telegram's encryption protocol, MTProto, is not secure, an... | Hacker News. Hacker News. URL: https://news.ycombinator.com/item?id=14375508 (date of access: 10.09.2023).
12. Telegram : Security vulnerabilities, CVEs. CVE security vulnerability database. Security vulnerabilities, exploits, references and more. URL: https://www.cvedetails.com/vulnerability-list/vendor_id-16210/Telegram.html (date of access: 29.11.2023).
13. Next Generation SSH2 Implementation. Elsevier, 2009. URL: https://doi.org/10.1016/b978-1-59749-283-6.x0001-3 (date of access: 29.11.2023).
14. Richards K. What is Cryptography? Definition from SearchSecurity. Security. URL: https://www.techtarget.com/searchsecurity/definition/cryptography (date of access: 29.11.2023).

15. Zhang W., Li S. Security measurements of steganographic systems. Applied cryptography and network security. Berlin, Heidelberg, 2004. P. 194–204. URL: https://doi.org/10.1007/978-3-540-24852-1_14 (date of access: 01.12.2023).
16. Information Hiding. (1st 1996 Cambridge, U.K.). Information hiding: First international workshop, Cambridge, U.K., May 30-June 1, 1996 : proceedings. Berlin : Springer, 1996. 350 p.
17. Литвин О. М., Першина Ю. І. Reconstruction of 3-D objects with use interflation of function. Оброблення сигналів і зображень та розпізнавання образів : Conf. on Automation, Control, and Information Technology, Новосибірськ. 2005.
18. Shehab D. A., Alhaddad M. J. Comprehensive Survey of Multimedia Steganalysis: Techniques, Evaluations, and Trends in Future Research. Symmetry. 2022. Vol. 14, no. 1. P. 117. URL: https://doi.org/10.3390/sym14010117 (date of access: 29.11.2023).
19. Кузнецов О. О., Євсеєв С. П., Король О. Г. СТЕГАНОГРАФІЯ : Навч. посіб. Харків : ХНЕУ, 2011. 207 с. URL: http://www.repository.hneu.edu.ua/jspui/handle/123456789/2289 (дата звернення: 07.11.2023).
20. Anderson R. J., Petitcolas F. A. P. On the limits of steganography. IEEE Journal on Selected Areas in Communications. 1998. Vol. 16, no. 4. P. 474–481. URL: https://doi.org/10.1109/49.668971 (date of access: 29.11.2023).
21. Nosrati M., Karimi R., Hariri M. An introduction to steganography methods. World Applied Programming. 2011. Vol. 1, no. 3. P. 191–195. URL: https://www.researchgate.net/profile/Masoud_Nosrati/publication/308646775_An_introduction_to_steganography_methods/links/57f41abc08ae280dd0b73d9f.pdf (date of access: 05.09.2023).
22. Amin Seyyedi S., Ivanov N. Statistical Image Classification for Image Steganographic Techniques. International Journal of Image, Graphics and

Signal Processing. 2014. Vol. 6, no. 8. P. 19–24. URL: https://doi.org/10.5815/ijigsp.2014.08.03 (date of access: 29.11.2023).

23. Pixinwav: residual steganography for hiding pixels in audio / M. Punti et al. ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing (ICASSP). 2022. P. 2485—2489. URL: https://arxiv.org/pdf/2106.09814.pdf (date of access: 10.04.2023).
24. A deep learning-based audio-in-image watermarking scheme / Das A., Zhong X. 2021 International Conference on Visual Communications and Image Processing (VCIP). 2021. P. 1-5. URL: https://arxiv.org/pdf/2101.01872v1.pdf (date of access: 10.04.2023).
25. Yang H. Cross-modal steganography: hiding video in audio : Doctoral dissertation. 2020. URL: https://repository.hkust.edu.hk/ir/Record/1783.1-109029 (date of access: 09.08.2023).
26. Beutelspacher A. Kryptologie: Eine Einführung in Die Wissenschaft Vom Verschlüsseln, Verbergen und Verheimlichen. Springer Vieweg. in Springer Fachmedien Wiesbaden GmbH, 2014.
27. Koshel T. ВИБІР МЕТОДА ЛІНГВІСТИЧНОЇ СТЕГАНОГРАФІЇ ДЛЯ ВИКОНАННЯ ЗАВДАНЬ З РЕЄСТРАЦІЇ СЕЙСМІЧНИХ ПОДІЙ. Системи управління, навігації та зв'язку. Збірник наукових праць. 2020. Т. 1, № 59. С. 79–85. URL: https://doi.org/10.26906/sunz.2020.1.079 (дата звернення: 29.11.2023).
28. Audio Steganography Techniques: A Survey / S. Mishra et al. Advances in Computer and Computational Sciences. Singapore, 2017. P. 581–589. URL: https://doi.org/10.1007/978-981-10-3773-3_56 (date of access: 29.11.2023).
29. Yadnya M. S., Kanata B., Anwar M. K. Using Phase Coding Method for Audio Steganography with the Stream Cipher Encrypt Technique. Proceedings of the First Mandalika International Multi-Conference on Science and Engineering 2022, MIMSE 2022 (Informatics and Computer

Science). Dordrecht, 2022. P. 66–75. URL: https://doi.org/10.2991/978-94-6463-084-8_8 (date of access: 29.11.2023).

30. Peak Signal-to-Noise Ratio as an Image Quality Metric. Mess- und Prüfsysteme, bei Emerson - NI. URL: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html (date of access: 29.11.2023).
31. Noor Azam M. H., Ridzuan F., Mohd Sayuti M. N. S. A New Method to Estimate Peak Signal to Noise Ratio for Least Significant Bit Modification Audio Steganography. Pertanika Journal of Science and Technology. 2022. Vol. 30, no. 1. P. 497–511. URL: https://doi.org/10.47836/pjst.30.1.27 (date of access: 29.11.2023).
32. Using Architecture Patterns to Architect and Analyze Systems of Systems / R. S. Kalawsky et al. Procedia Computer Science. 2013. Vol. 16. P. 283–292. URL: https://doi.org/10.1016/j.procs.2013.01.030 (date of access: 29.11.2023).
33. Farshidi S., Jansen S., van der Werf J. M. Capturing software architecture knowledge for pattern-driven design. Journal of Systems and Software. 2020. Vol. 169. P. 110714. URL: https://doi.org/10.1016/j.jss.2020.110714 (date of access: 29.11.2023).
34. Rana M. E., Saleh O. S. High assurance software architecture and design. System Assurances. 2022. P. 271–285. URL: https://doi.org/10.1016/b978-0-323-90240-3.00015-1 (date of access: 29.11.2023).
35. System Assurances. Elsevier, 2022. URL: https://doi.org/10.1016/c2020-0-03092-6 (date of access: 29.11.2023).
36. What is Component-Based Architecture?. Mendix. URL: https://www.mendix.com/blog/what-is-component-based-architecture (date of access: 29.11.2023).

37. Component-Based Architecture. Online Tutorials, Courses, and eBooks Library | Tutorialspoint. URL: https://www.tutorialspoint.com/software_architecture_design/component_based_architecture.htm (date of access: 29.11.2023).
38. Bots: An introduction for developers. Telegram APIs. URL: https://core.telegram.org/bots (date of access: 29.11.2023).
39. Introduction to Python. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/python/python_intro.asp (date of access: 29.11.2023).
40. Python Tutorial | Learn Python Programming. GeeksforGeeks. URL: https://www.geeksforgeeks.org/python-programming-language (date of access: 29.11.2023).
41. What Is Python Used For? A Beginnerâs Guide. Coursera. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (date of access: 29.11.2023).
42. Ruby Programming Language. Ruby Programming Language. URL: https://www.ruby-lang.org/en/ (date of access: 29.11.2023).
43. About Ruby. Ruby Programming Language. URL: https://www.ruby-lang.org/en/about (date of access: 29.11.2023).
44. JavaScript | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (date of access: 29.11.2023).
45. Top 10 Python Libraries to Create Your Telegram Bot Easily (GitHub). Be on the Right Side of Change. URL: https://blog.finxter.com/top-10-python-libraries-to-create-your-telegram-bot-easily-github (date of access: 29.11.2023).
46. Python framework for Telegram bots. botogram. URL: https://botogram.dev (date of access: 29.11.2023).
47. Khaliullin R. Robert Khaliullin on LinkedIn: #python #aiogram #pytelegrambotapi #telepot #telebot #pytelegrambotapi2… LinkedIn: Log

In or Sign Up. URL: https://www.linkedin.com/posts/robert-khaliullin_python-aiogram-pytelegrambotapi-activity-7070990445513711616-oVuv (date of access: 29.11.2023).

48. About us. Hetzner, Dedicated Server, Cloud, Storage & Hosting. URL: https://www.hetzner.com/unternehmen/ueber-uns (date of access: 29.11.2023).
49. Honepage. Hetzner, Dedicated Server, Cloud, Storage & Hosting. URL: https://www.hetzner.com/ (date of access: 29.11.2023).
50. Overview - Hetzner Docs. Hetzner Docs. URL: https://docs.hetzner.com/cloud/servers/overview/ (date of access: 29.11.2023).
51. Features - PyCharm. JetBrains. URL: https://www.jetbrains.com/pycharm/features (date of access: 29.11.2023).
52. Built-in developer tools - features | pycharm. JetBrains. URL: https://www.jetbrains.com/pycharm/features/tools.html (date of access: 29.11.2023).
53. Full-stack web development - features | pycharm. JetBrains. URL: https://www.jetbrains.com/pycharm/features/web_development.html (date of access: 29.11.2023).
54. CENTAURI | bit error rate | what is A good BER. Transcelestial Support. URL: https://support.transcelestial.com/support/solutions/articles/51000324695-bit-error-rates-what-is-ber-and-what-is-a-good-ber- (date of access: 29.11.2023).
55. Bit error rate analysis techniques- MATLAB & simulink. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/help/comm/ug/bit-error-rate-analysis-techniques.html (date of access: 29.11.2023).

56. Analyze BER performance of communications systems - MATLAB. MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink. URL: https://www.mathworks.com/help/comm/ref/biterrorrateanalysis-app.html (date of access: 29.11.2023).
57. Information hiding in audio steganography using LSB matching revisited / R. Shanthakumari et al. Journal of physics: conference series. 2021. Vol. 1911, no. 1. P. 012027. URL: https://doi.org/10.1088/1742-6596/1911/1/012027 (date of access: 29.11.2023).

ДОДАТОК А

Програмна реалізація PSNR тесту для вхідного та зашумленого WAV файлу на мові програмування Python.

```
from scipy.io import wavfile
from scipy.io.wavfile import WavFileWarning
from math import log10, sqrt
import cv2
import numpy as np

def PSNR(original, noise):
    # Crop noise to same length as original
    noise = noise[:len(original)]
    mse = np.mean((original - noise) ** 2)
    if(mse == 0):  # MSE is zero means no noise is present in the signal .
                # Therefore PSNR have no importance.
        return 100
    max_db = 65536
    psnr = 20 * log10(max_db / sqrt(mse))
    return psnr


import warnings


warnings.filterwarnings("ignore", category=WavFileWarning)

originalsample, original = wavfile.read('1.wav')

for i in range(1, 11):
    noisesample, noise = wavfile.read(f'ouf_{i}.wav')
    value = PSNR(original, noise)
    print(f"PSNR value for output_container{i}.wav is {value} dB")
```

Рисунок А.1 - Скрипт для реалізації PSNR тесту