МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»

Завідувач кафедри
_____ Жуковицький І. В.
(підпис) (ПІБ)
«21» 12 2021 р.

ДИПЛОМНА РОБОТА
на здобуття освітнього ступеня «магістр»

Галузь знань 12 Інформаційні технології
(шифр) (назва)

Спеціальність 123 Комп'ютерна інженерія
(код) (повна назва)

Тема Дослідження та розробка засобів захищеного обміну повідомленнями

Theme Research and development of secure messaging means

Керівник дипломної роботи доцент _____ Остапець Д. О.
(посада) (підпис) (ПІБ)

Консультант розділу з БЖД доцент _____ Саблін О. І.
(посада) (підпис) (ПІБ)

Нормоконтролер доцент _____ Шаповалов В. О.
(посада) (підпис) (ПІБ)

Студент групи КС2021 _____ Любушкін Д. Є.
(група) (підпис) (ПІБ)

Student Liubushkin Denys
(family name)

Дніпро
2021

Міністерство освіти і науки України

Український державний університет науки і технологій

Кафедра «Електронні обчислювальні машини»

ДОВІДКА

За результатами перевірки випускної кваліфікаційної роботи здобувача вищої освіти Любушкіна Дениса Євгеновича

на тему:

«Дослідження та розробка засобів захищеного обміну повідомленнями»

в роботі не виявлено порушень академічної доброчесності.

Керівник ВКР, к.т.н., доц. каф. ЕОМ **Д.О. Остапець**

Український державний університет науки і технологій

Факультет Комп'ютерних технологій і систем кафедра Електронні обчислювальні машини
Спеціальність Комп'ютерна інженерія

«ЗАТВЕРДЖУЮ»
Завідувач кафедри

_____
(підпис)
«___»_____20_ р.

ЗАВДАННЯ

до дипломної роботи на здобуття освітнього ступеня ___«магістр»_____
(освітнього ступеня)

студента групи КС2021 _____Любушкіна Дениса Євгеновича_____
(номер групи) (ПІБ)

1 Тема дипломної роботи Дослідження та розробка системи захищеного обміну повідомленнями _____
_____
_____

затверджена наказом по університету від «31»__08__2021_ р. № 508ст__.

2 Термін подання студентом закінченої роботи_____

3 Вихідні дані до дипломної роботи_____
_____
_____
_____

4 Зміст пояснювальної записки (перелік питань до розробки) _____
_____
_____
_____
_____

5 Перелік креслень (демонстраційного матеріалу) _____
_____
_____
_____
_____
_____
_____
_____
_____

6 Розділи та консультанти

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека у надзичайних ситуаціях	Саблін О. І.		

КАЛЕНДАРНИЙ ПЛАН

Назва розділу	Термін виконання	Обсяг розділу, %

Дата видачі завдання: «___»_____ 20___ р.

Керівник дипломної роботи _____ Остапець Д. О.
(підпис) (ПІБ)

Завдання прийняв до виконання _____ Любушкін Д. Є.
(підпис) (ПІБ)

РЕФЕРАТ

Любушкін Д. Є. Дослідження та розробка засобів захищеного обміну повідомленнями. – Український державний університет науки і технологій, кафедра електронних обчислювальних машин. – Дипломна робота. – 81с., 27 рис., 5 табл., 21 джерело, 2 додатки.

Об’єктом дослідження в даній дипломній роботі є засоби захищеного обміну повідомленнями. Метою дипломної роботи є розробка програмних засобів захищеного обміну повідомленнями, в яких передбачено використання раціонального набору криптографічних функцій з точки зору збереження конфіденційності повідомлення та швидкодії апаратної частини. Проведено порівняльний аналіз перечислених засобів, їх функціональних частин, виділено їх переваги і недоліки у підходах до захисту інформації. Розроблено архітектуру власного комплексу засобів, його структуру, алгоритми роботи, розроблено та налагоджено програмне забезпечення. Проведено дослідження залежності часу виконання криптографічних операцій від довжини передаваного повідомлення для різного складу апаратної частини. Наведено рекомендації з використання окремих криптографічних алгоритмів для різних умов. Складено інструкцію із використання засобів та методику їх використання у навчальному процесі.

МЕСЕНДЖЕР, ОБМІН ПОВІДОМЛЕННЯМИ, КРИПТОГРАФІЧНИЙ АЛГОРИТМ, АУТЕНТИФІКАЦІЯ, КОНФІДЕНЦІЙНІСТЬ, ЗАХИСТ ДАНИХ, AES, RC4, SHA, PYTHON, SQLite.

ABSTRACT

Liubushkin D. Research and development of secure messaging means – Ukrainian State University of Science and Technology, department of electronic computing machines. – Diploma project. – 83 p., 27 fig., 6 tables, 21 sources, 2 appendixes.

The subject of research in this project is means of secure message exchange. The goal of this project is the development of software means of secure message exchange, which provides the usage of rational set of cryptographic functions from point of view of hardware performance. An overview and analysis of already existing means of secure messaging and security mechanisms that they use was performed. A comparative analysis of said means, their functionality and their advantages and disadvantages was carried out. The architecture of own complex of means, its structure, algorithms of work has been developed, the software was developed and debugged. A study of the dependence of the time of cryptographic operations on the length of the transmitted message for different hardware. Recommendations for the use of separate cryptographic algorithms for different conditions are given. The instruction on use of means and a technique of their use in educational process was developed.

MESSENGER, MESSAGE EXCHANGE, CRYPTOGRAPHIC ALGORITHM, AUTHENTICATION, CONFIDENTIALITY, DATA SECURITY, AES, RC4, SHA, PYTHON, SQLite.

ЗМІСТ

ВСТУП

Месенджери з кожним роком все глибше проникають у наше повсякденне життя, дозволяючи обмінюватись миттєвими повідомленнями з іншими людьми. І разом з цим зростає і кількість кіберзлочинів, тому питання конфіденційності передачі даних мережею Internet стоїть надзвичайно гостро, адже наслідки таких злочинів можуть бути тяжкими як для фізичної, так і для юридичної особи. Постраждати від такого злочина може як репутація, так і життя, тому розглядуване питання не тільки не втрачає актуальності, а лише стає все більш актуальним, отже і тема роботи є актуальною.

Тема дипломної роботи затверджена наказом по університету №508ст від 31.08.2021 р.

Метою дипломної роботи є розробка програмних засобів захищеного обміну повідомленнями, в яких передбачено використання раціонального набору криптографічних функцій з точки зору збереження конфіденційності повідомлення та швидкодії апаратної частини.

Основні положення роботи доповідались та були схвалені на XIV та XV Міжнародних науково-практичних конференціях «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» у 2020-2021 рр.

Представлена дипломна робота складається зі вступу, 6 розділів та висновків.

У розділі 1 представлений огляд існуючих засобів обміну повідомленнями та їх інструментів захисту інформації в них. Наведено відмінні риси у функціональних частинах та механізмах захисту, виділено переваги та недоліки того чи іншого підходу.

У розділі 2 наведено організацію комплексу та розроблено його інформаційну структуру. Наведено опис процесів взаємодії між клієнтською та серверною частинами комплексу.

У розділі 3 здійснено вибір мови програмування та середовища для розробки програмного забезпечення, описано функції та алгоритми їх роботи, на основі яких розроблено програмне забезпечення.

У розділі 4 проведено дослідження часу виконання криптографічних операцій від довжини шифрованого повідомлення на прикладі двох електронно обчислювальних машин різної обчислювальної потужності та двох криптоалгоритмів: блокового та потокового. Дані проаналізовано та наведено у вигляді таблиць та графіків.

У розділі 5 наведено методику використання розробленого комплексу засобів, зокрема у навчальних цілях.

У розділі 6 сформовано основні вимоги безпеки при виконанні робіт із електронно обчислювальною машиною. Розглянуто найпоширеніші шкідливі виробничі фактори, характерні для такого виду роботи. Наведено порядок дій працівника у випадку надзвичайних ситуацій.

У додатках А, Б наведено вихідний код програм.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ СИСТЕМ ОБМІНУ ПОВІДОМЛЕННЯМИ

1.1 Загальні відомості

Основним методом комунікації між людьми у мережі Інтернет є месенджери, які дозволяють швидко обмінюватись текстовими повідомленнями, аудіо- та відеофайлами і мати змогу дізнатись про те, що адресат ознайомився із повідомленням.

Месенджер – програмний засіб, який дозволяє двом або більше клієнтам обмінюватись миттєвими повідомленнями через мережу Інтернет.

Клієнт – програмне забезпечення на пристрої користувача, що дозволяє обмінюватись повідомленнями із іншими користувачами через сервер.

Сервер – апаратне забезпечення, яке призначене для збереження, отримання, обробки та передачі даних, отриманих від клієнтів та подальшого їх відправлення клієнтам.

Ідентифікація – процес розпізнання суб’єкта системою за допомогою деякою унікальної характеристики.

Аутентифікація – процес підтвердження своєї особистості перед системою з обмеженим доступом.

Шифрування – процес оборотного перетворення інформації із викривленням її змісту з метою уникнення ознайомлення з нею несанкціонованих осіб або зловмисників.

Дешифрування – процес відновлення викривленої шифруванням інформації до її початкового стану із початковим об’ємом та змістом.

Наскрізне шифрування – такий вид шифрування інформації, за якого інформація шифрується на стороні відправника і розшифровується лише на стороні отримувача.

Хеш – інформація, що є результатом роботи односторонньої функції, а саме хеш-функції

Конфіденційність – характеристика інформації, що означає збережену секретність, нерозкриття інформації для несанкціонованих осіб

Цілісність – характеристика інформації, означає, що інформація зберегла свій початковий вигляд і не піддавалась змінам несанкціонованими особами

Доступність – характеристика інформації, означає, що існує можливість ознайомитись із інформацією тим особам, у яких передбачено до неї доступ

1.2. Огляд популярних месенджерів

На даний момент найбільш відомими месенджерами за думкою автора постають Telegram, Viber, WhatsApp та Skype. Також пропонується розглянути месенджер Signal, який позиціонується як захищений, та месенджер Tox, який ще знаходиться на стадії розробки і має на меті стати повністю децентралізованим.

Рисунок 1.1 - Найбільш популярні месенджери

1.2.1 Telegram [1]

Telegram це програма обміну повідомленнями з відкритим кодом, розроблена компаніями Telegram FZ LLC та Telegram Messenger Inc. Реєстрація акаунту, авторизація та аутентифікація у програмі здійснюється за допомогою номеру мобільного телефону. Опціонально можна додати другий фактор захисту облікового запису за допомогою PIN-коду, який потрібно вводити для аутентифікації користувача.

Ця програма дозволяє обмінюватись текстовими, аудіо-, відеоповідомленнями і файлами, а також здійснювати дзвінки та відеодзвінки. Існує можливість обмінюватись повідомленнями із використанням наскрізного (end-to-end) шифрування даних, але це потребує додаткових дій зі сторони користувача. Також Telegram є централізованим, тому може існувати ймовірність того, що тримач серверу може отримати доступ до повідомлень користувача.

Шифрування у Telegram здійснюється за допомогою власної розробки Telegram LLC - протоколу MTProto, обмін ключами здійснюється за протоколом Діфі-Хелмана.

Telegram зберігає повідомлення і медіа-файли на хмарному сервері у випадку звичайних чатів, у випадку ж секретних чатів, для яких обов'язкове наскрізне шифрування, сервер виконує лише функцію посередника, через якого два клієнти обмінюються повідомленнями та ключами шифрування для них. Через те, що сервер не зберігає повідомлення із секретних чатів, їх синхронізація між пристроями користувача неможлива.

Після створення двома клієнтами спільного секретного ключа, клієнту А для відправки повідомлення потрібно зашифрувати його спільним ключем та об'єднати зашифроване повідомлення із хешованим ключем повідомлення, який складається із метаданих цього повідомлення. Ключ повідомлення потрібний для веріфікації клієнтом Б факту невтручання третіх особ у процес листування.

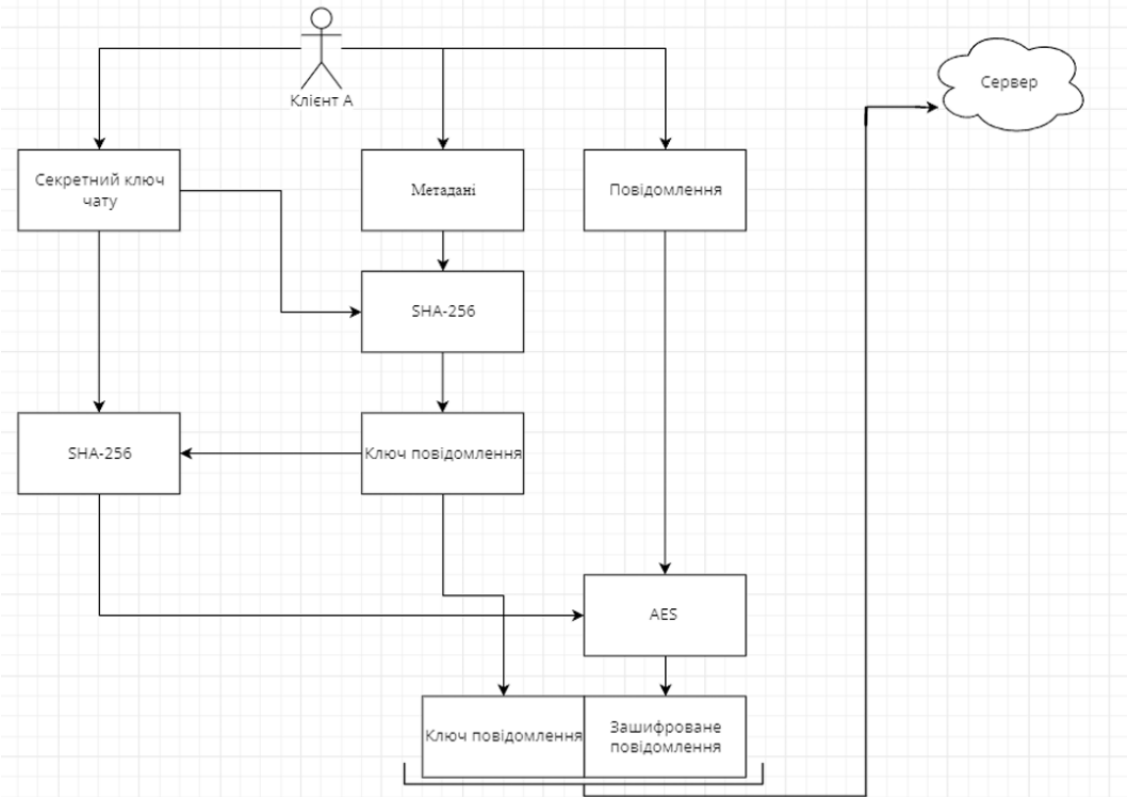

Рисунок 1.2 - Схема шифрування даних клієнтом для відправки на сервер

1.2.2 Whatsapp [2]

WhatsApp це месенджер із закритим кодом, якою володіє компанія Facebook Inc. Реєстрація акаунту, авторизація та аутентифікація у програмі здійснюється за допомогою номеру мобільного телефону. Як і у Telegram, існує можливість захистити акаунт додатковим PIN-кодом для аутентифікації.

Програма дозволяє обмінюватись файлами, текстовими та аудіоповідомленнями, а також здійснювати дзвінки та відеодзвінки.

На відміну від Telegram, наскрізне шифрування у WhatsApp працює для будь-якого чату, не потребуючи жодних дій зі сторони користувача, однак, на відміну від Telegram, у Whatsapp присутня функція синхронізації повідомлень між пристроями користувача, ця можливість присутня завдяки зашифрованим резервним копіям, що зберігаються на сервері. Месенджер, як і Telegram, є

централізованою системою. Шифрування здійснюється за алгоритмом Curve25519.

1.2.3 Viber [3]

Viber – програма обміну повідомленнями із закритим кодом, яким володіє компанія Rakuten. Реєстрація, авторизація та аутентифікація здійснюються за допомогою номеру мобільного телефону, також існує можливість сховати чат із певними людьми, встановивши PIN-код для чату із певною людиною, в такому випадку чат буде відображуватись у програмі лише при введенні PIN-коду для відповідної людини у стрічку пошуку контактів.

Viber дозволяє обмінюватись текстовими, аудіо- та відеоповідомленнями і здійснювати аудіо- та відеодзвінки. Як і у WhatsApp, присутнє наскрізне шифрування повідомлень, яке увімкнене завжди та не потребує втручання користувача. Наскрізне шифрування працює лише за умови, що кожен отримувач і відправник користуються останньою версією програми.

Viber не зберігає історію листування між клієнтами на своїх серверах, тому, у випадку видалення додатку з усіх пристроїв користувача, неможливо буде відновити повідомленнями, якими користувач обмінювався з іншими користувачами. У Viber присутня можливість зробити резервну копію історії листування та відновити ці дані тому ж пристрої у разі видалення додатку або зовсім на іншому пристрої, що однак підвищує ризик порушення конфіденційності, якщо цієї резервної копії дістанеться зловмисник.

Месенджер є централізованим. Шифрування повідомлень здійснюється за алгоритмом Salsa20. Існує можливість відправляти повідомлення, які

самоліквідуються через заданий час. Такі повідомлення використовують наскрізне шифрування.

1.2.4 Skype [4]

Skype – централізований месенджер із закритим кодом для здійснення аудіо- та відеоконференцій, але також присутній функціонал обміну текстовими повідомленнями, аудіоповідомленнями і файлами. Розроблена компанією Skype Technologies, наразі компанія Microsoft володіє Skype.

У Skype присутня можливість наскрізного шифрування, однак працює вона лише для обміну повідомленнями тет-а-тет і не працює у групових чатах або дзвінках. Для активації наскрізного шифрування користувача потрібно увімкнути її у меню налаштування, за умовчанням вона вимкнена і працює лише за умови використання відправником та отримувачем останньої версії Skype.

Для реєстрації і авторизації використовується електронна пошта або номер мобільного телефону, для аутентифікації використовується пароль. Для захисту конференцій можна встановити кожному з учасників унікальний PIN-код для доступу до конференції.

Шифрування здійснюється за алгоритмом AES із 256-бітними ключами, публічні ключі користувачів шифруються за допомогою 1536 або 2048-бітних ключів RSA.

1.2.5 Tox Project [5]

Tox Project – система обміну повідомленнями з відкритим кодом, що використовує протокол Tox. Розроблена і підтримується ентузіастами. Не потребує для реєстрації ані адреси електронної пошти, ані номеру мобільного телефону, що є як і перевагою (якщо буде вкрадено пошту чи телефон, неможливо відновити обліковий запис), так і недоліком (якщо пароль було забуто, то доступ до акаунту втрачено назавжди). Серед можливостей Tox є

обмін текстовими повідомленнями, файлами та здійснення аудіо- та відеодзвінків.

На відміну від Telegram, Viber, WhatsApp, Skype та Signal, клієнти Tox доступні лише на Windows, GNU/Linux, BSD, MacOS та Android. Серед відмінностей Tox можна виділити наскрізне шифрування за умовчанням та наближеність до децентралізованості. Протокол Tox може використовувати користувачів мережі Tox для передачі повідомлень замість центрального серверу, однак повністю виключити центральний сервер неможливо: він потрібен для реєстрації імен користувачів у системі.

Обмін ключами здійснюється за допомогою алгоритму curve25519, який є різновидом протоколу Діфі-Хелмана, симетричне шифрування працює за алгоритмом xsalsa20. Аутентифікація користувача відбувається за допомогою паролю.

1.2.6 Signal [6]

Signal – програма обміну повідомленнями із відкритим кодом, розроблена компаніями Signal Technology Foundation та Signal Messenger LLC.

Для реєстрації, авторизації та аутентифікації використвується номер мобільного телефону. Додатково можна захистити акаунт PIN-кодом, який потрібно вводити при вході у додаток.

Присутні можливості текстових та аудіоповідомлень, обміну файлами і здійснення аудіо- та відеодзвінків.

Шифрування здійснюється за алгоритмом Double Ratchet. Наскрізне шифрування завжди увімкнене без потреби яких-небудь дій зі сторони користувача.

1.3 Порівняльна характеристика месенджерів

Проведемо аналіз наявного в месенджерах функціоналу та механізмів захисту інформації. Результати зведено в таблиці 1.1 і таблиці 1.2.

Таблиця 1.1 – Функціонал месенджерів

Месенджер / Функція	Telegram	WhatsApp	Viber	Skype	Tox	Signal
Текстові повідомлення	+	+	+	+	+	+
Голосові повідомлення	+	+	+	+	-	+
Відеоповідомлення	+	-	+	-	-	-
Медіа-файли	+	+	+	+	+	+
Дзвінки	+	+	+	+	+	+

Таблиця 1.2 – Порівняльна характеристика захисних механізмів месенджерів

Месенджер / Механізм захисту	Telegram	WhatsApp	Viber	Skype	Tox	Signal
Наявність наскрізного шифрування	+	+	+	+	+	+
Обов'язковість наскрізного шифрування	-	+	+	-	+	+
Захист PIN-кодом або паролем	+	+	+	+	-	+
Децентралізованість	-	-	-	-	розподіленість	-
Наявність відкритого коду	+	-	-	-	+	+

В результаті проведеного огляду існуючих систем обміну повідомленнями було виявлено, що ідеального месенджеру, який би поєднував можливість забезпечення цілісності і доступності даних користувача із надійним захистом цих даних. Кожен із існуючих продуктів мав іти на компроміс: або дати користувачеві змогу відновити свої дані, тим самим підвищивши ризик перехоплення даних користувача третьою особою, або ж унеможливити використання одного і того ж чату на різних пристроях заради того, щоб створювані резервні копії не могли скомпрометувати результати роботи наскрізного шифрування листування між двома користувачами.

Небезпеку у процесі листуванні може становити присутність серверу, якому користувач має довіряти у тому, що той не буде зловживати оброблюваною ним інформацією, тобто існує сценарій, за якого той, хто володіє сервером, може використати збережені дані користувачів у власних цілях: наприклад продати або передати третім особам, що може становити небезпеку для безпосередніх користувачів.

Месенджер Signal намагається поєднувати у собі всі переваги зручності використання і безпечності листування, тоді коли Telegram, WhatsApp та Viber більш віддають перевагу зручності, підвищуючи ризик порушення конфіденційності процесу листування.

Найбільш наближеним же до децентралізації виявився мессенджер Tox. Оскільки створити повністю децентралізований месенджер неможливо, Tox пішов дорогою розподіленості. Тобто центр (сервер) потрібен для реєстрації, ідентифікації та аутентифікації користувачів, після чого трафік може транслюватись через уже існуючих користувачів і роль сервера у процессі листування значно менша ніж у інших зазначених месенджерах. Але таке рішення також не є безпечним, замість знищення посередника, воно створює багато випадкових посередників, які можуть спробувати перехопити дані листування. Тому рішення замінити одного посередника на багатьох посередників може виявлятись сумнівним.

1.4 Висновки за розділом

В ході першого розділу було виконано огляд існуючих рішень на ринку месенджерів, у тому числі тих, що позиціонуються як захищені. Були розглянуті механізми захисту у наявних системах обміну повідомленнями, які мають дозволяти користувачеві безпечно спілкуватись із іншими користувачами сервісу, та їх функціональні частини, які покращують

зручність використання месенджера та надають змогу обмінюватись різними видами інформації.

За результатами огляду складено таблиці із порівняльними характеристиками функціональних частин та захисних механізмів у зазначених месенджерах.

В ході проведеного аналізу було виявлено, що у процесі користування месенджером за різних причин можуть виникати загрози для цілісності, доступності та конфіденційності інформації у листуванні.

2 ОРГАНІЗАЦІЯ РОЗРОБЛЮВАНОГО КОМПЛЕКСУ ЗАСОБІВ ЗАХИЩЕНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

2.1 Процес обміну даними між абонентами

Для здійснення обміну повідомленнями дві сторони (два клієнти), між якими і буде відбуватись процес листування, та третя сторона (сервер), через який здійснюватиметься обмін ключами за власною реалізацією протоколу обміну ключів Діфі-Хелмана[7] (схему розробленої реалізації протоколу наведено на рисунку 3.1) між клієнтами та передача шифрованих повідомлень від одного клієнта до іншого. Також сервер зберігатиме повідомлення та ключі клієнтів до моменту завершення передачі їх адресату, після цього сервер видалить цю інформацію задля безпеки користувачів.

Для передачі інформації використовуватиметься протокол передачі даних TCP. Серед переваг цього протоколу можна виділити надійність – у протоколі наявна функція підтвердження прийому, тобто якщо дані не дійшли до отримувача, вони відправляються повторно, на відміну від протоколу UDP, у якому дані можуть не дістатись отримувача. Також перевагою протоколу є той факт, що дані у пакетах приходять у тому порядку, в якому були відправлені, оскільки усі сегменти TCP пакетів помічаються номером послідовності у заголовку.

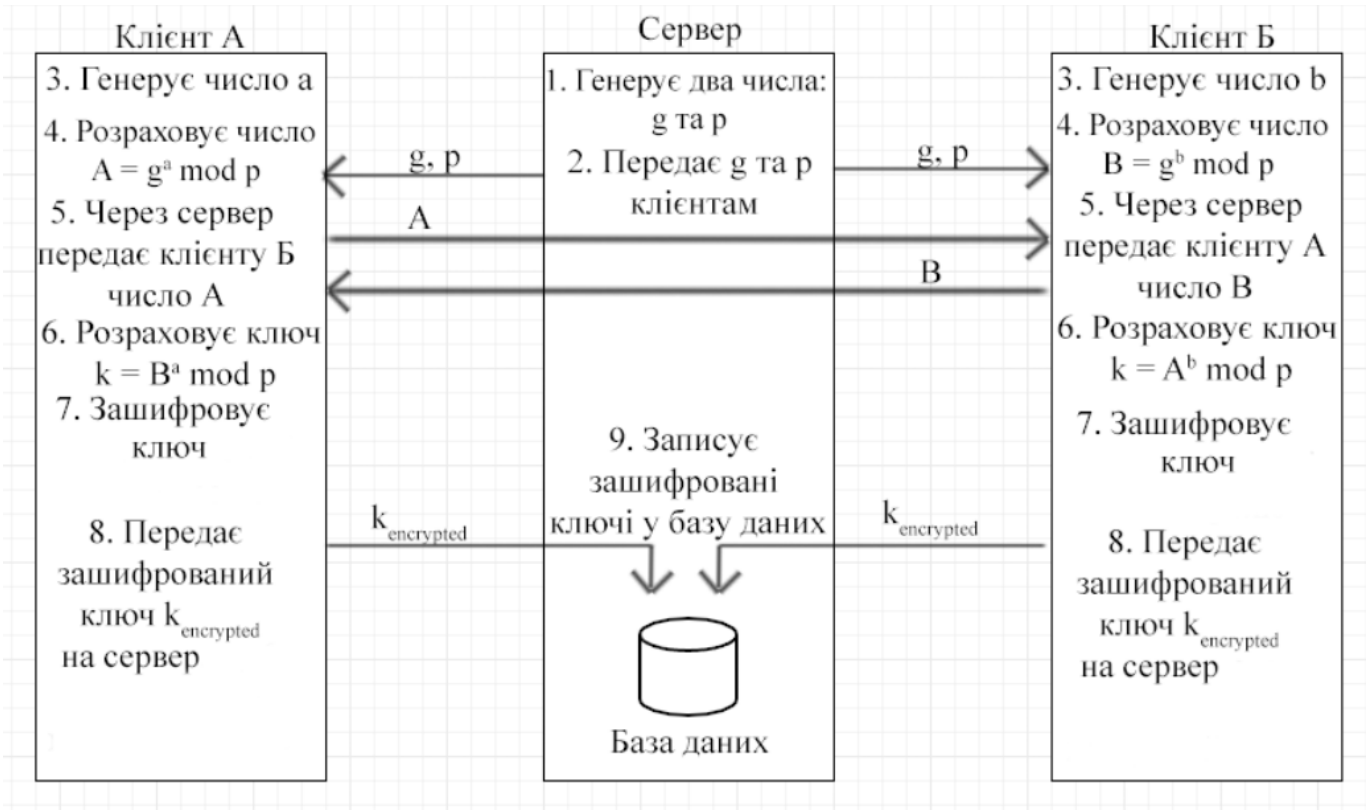

Рисунок 2.1 – Схема обміну ключами між клієнтами

Для усунення можливості ідентифікувати користувача за номером телефону пропонується використовувати для ідентифікації виданий адміністратором користувачеві логін.

Наявність серверу виправдана тим, що він виконуватиме функцію зберігання історії листування між користувачами, забезпечувати реєстрацію, ідентифікацію, аутентифікацію користувачів та обмін ключами.

Оскільки у базі даних буде передбачено зберігання історії листування користувачів, то повідомлення у клієнті користувача відображуватимуться по запиту до серверу. Схема спілкування виглядатиме приблизно так: користувач А пише повідомлення, клієнт шифрує повідомлення та відправляє його на сервер, на сервері шифроване повідомлення додається до бази даних і сервер повідомляє отримувача повідомлення про наявність нового повідомлення для нього, після чого клієнт отримувача виконує запит до бази даних сервера і

отримує нове повідомлення, яке відображується для отримувача у вікні клієнта. Схему наведено на рисунку 3.2. Схема припускає, що обмін ключами чи отримання вже створених ключів із бази даних клієнтами вже виконано.

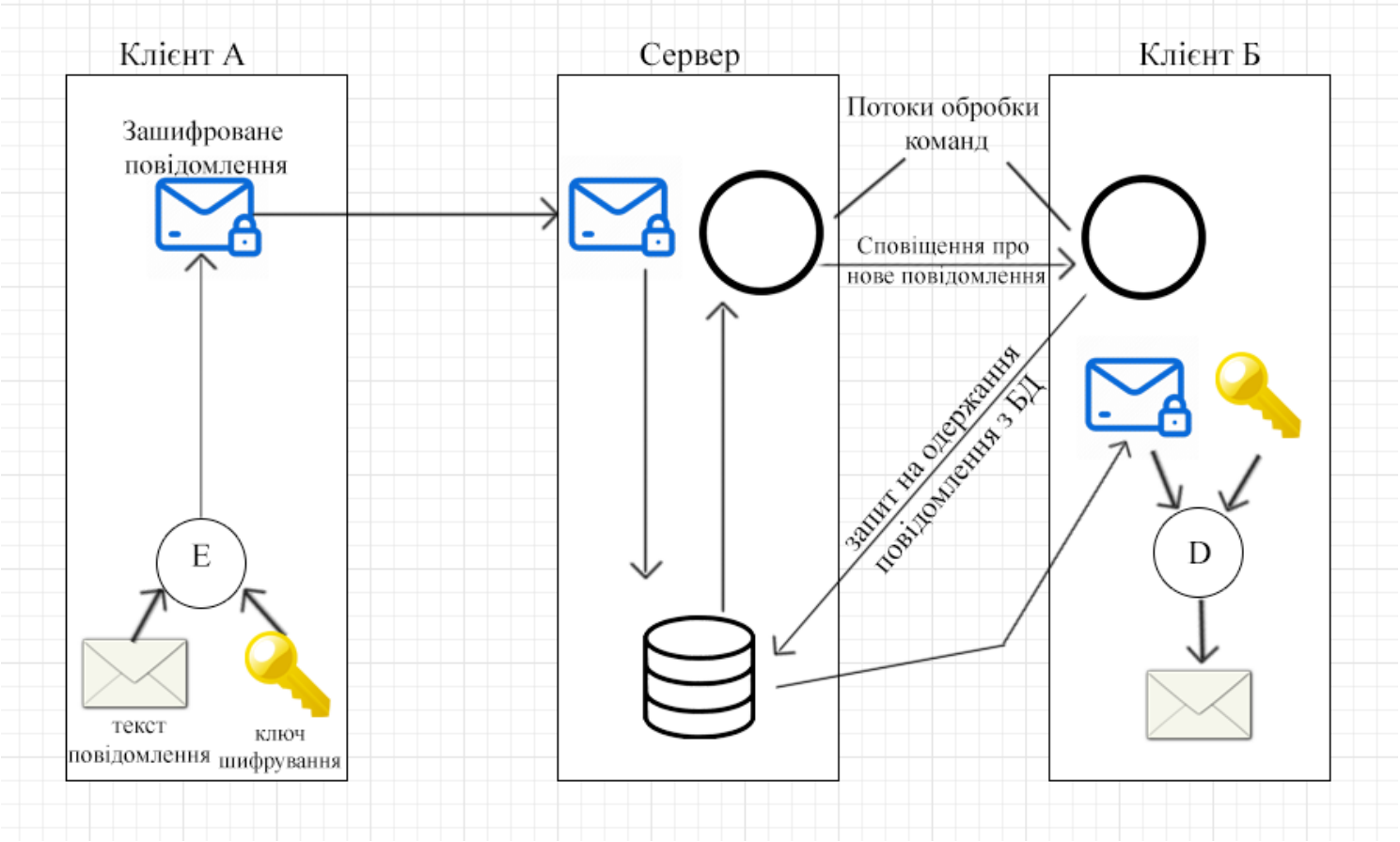

Рисунок 2.2. – Схема відправки та отримання повідомлення

2.2 Процес ідентифікації та аутентифікації користувача

Для реєстрації використовуватиметься виданий адміністратором користувачеві логін та пароль. Паролі користувачів зберігатимуться у базі даних серверу.

Зберігання паролів у чистому вигляді – дуже небезпечна практика, адже у випадку витоку даних із бази даних сервера, усі дані для аутентифікації користувачів буде скомпрометовано. Саме для цього знадобиться хеш-функція. За думкою автора найоптимальнішим варіантом хеш-функції буде

функція SHA-256. Серед інших хеш-функцій SHA-256 вигідно виділяється невеликою кількістю колізій.

Вхідний оригінальний текст для хеш-функції SHA-256 не має перевищувати 264 біт для того, щоб алгоритм міг видати найбільш випадкове хеш-значення. Довжина хеш-дайджесту дорівнюватиме 256 біт. Результат хеш-функції є необоротним.

Маючи на руках хеш паролю користувача, зловмисник не зможе аутентифікуватись від особи користувача, оскільки для входу у систему потрібний саме пароль, з якого було створено цей хеш.

2.3 Організація листування і збереження даних

Для реалізації збереження доступності даних пропонується зберігати історію листування у базі даних серверу в зашифрованому вигляді. Процес обміну повідомленнями виглядатиме наступним чином: абонент А пише повідомлення для абонента Б, програма-клієнт абонента А шифрує повідомлення, зашифроване повідомлення відправляється на сервер і зберігається у базі даних, після чого надсилає програмі-клієнту абонента Б сигнал про те, що для нього є нове повідомлення від абонента А. Тоді, якщо у клієнті Б відкрито чат із абонентом А, то він надсилає серверу сигнал оновлення історії листування із бази даних, сервер надсилає клієнту Б історію листування із абонентом А, клієнт Б її розшифровує і відображує у вікні додатка.

Із ціллю зберігання даних про користувачів та повідомлення потрібно розробити структуру бази даних. Для роботи додатку знадобиться три таблиці: таблиця із даними аутентифікації користувачів, що включатиме в себе поле логіна та поле хешу пароля.

У другій таблиці зберігатимуться записи про шифровані ключі для зашифрування та розшифрування повідомлень у діалогах із іншими

користувачами. Полями цієї таблиці будуть «Власник ключа», «Співбесідник, для якого цей ключ потрібен» і сам «Ключ».

В третій таблиці планується зберігати записи про повідомлення. Ця таблиця зберігатиме дані про відправника, отримувача, зашифрований текст повідомлення, час його відправлення, час прочитання. Час отримання необхідний для того, щоб було зрозуміло, чи було прочитаним повідомлення.

Схему бази даних наведено на рисунку 3.3.

Таблиця Users	
Username	Password
Логін користувача	Шістнадцятирічний дайджест хешу пароля користувача

Таблиця Messages				
Sender	Recipient	Text	Sent_time	Seen_time
Логін відправника	Логін отримувача	Текст повідомлення	Час відправлення	Час прочитання

Таблиця Keys		
Owner	For_user	Key
Власник ключа	Користувач, для якого використовуватиметься цей ключ	Зашифрований ключ

Рисунок 2.3. – Структура бази даних додатка

Для шифрування ключів і листування вирішено обрати криптоалгоритм AES-256 із довжиною блоку 128 біт і ключем довжиною 256 біт[8]. Причини такого рішення наступні:

- Для взлому шифру AES із ключем довжиною 256 біт потрібно 2^{128} спроб, що займе достатньо великий проміжок часу навіть із сучасною обчислювальною технікою
- Він є одним із найпоширеніших шифрів у сфері комерційних рішень, рішень із зберігання даних, проведення електронних транзакцій тощо
- Шифр AES досі є стандартом прийнятим Національним Інститутом Стандартів та Технологій (NIST) ще з 2001 року

Криптоалгоритм AES надає достатній рівень захисту для того, щоб обмін повідомленнями у розроблюваному засобі вважався захищеним.

2.4 Можливі загрози системі

У процесі обміну даними мережею Інтернет потрібно враховувати, що використовується незахищений канал передачі даних. Саме з метою захисту даних під час їх передачі і використовуються такі методи як шифрування і хешування.

У розроблюваному засобі не передбачено передачі даних у відкритому вигляді, тому ще це вкрай небезпечна процедура. Паролі передаються у хешованому вигляді, повідомлення шифруються спільним ключем шифрування, котрий користувачі обчислюють без необхідності його явної передачі один одному, а ключі шифрування – за необхідності їх передачі – передаються у шифрованому вигляді персональним ключем шифрування, який відомий лише клієнту.

Однак подібні рішення не виключають усіх факторів ризику, що несе із собою незахищений канал передачі даних. Зловмисник може проводити так звану атаку «Man in the Middle»[9] (атака «людина посередині») - прослуховувати трафік, що проходить мережею, за допомогою таких інструментів як, наприклад, Wireshark. Пізніше він може проаналізувати цей трафік і намагатись зламати шифр, і, хоча система є достатньо захищеною, це

не означає, що цей фактор загрози можна ігнорувати. Для захисту трафіка можна використовувати віртуальну приватну мережу, більш відому під абревіатурою VPN (Virtual Private Network). VPN допоможе замаскувати вашу реальну IP-адресу, вашу локацію та надасть додатковий шар шифрування усім даним, що передаються та приймаються засобом.

Також існує ймовірність підміни IP-адреси, здійснюється це за допомогою створення і відправки IP-пакетів зі сфальсифікаваною IP-адресою джерела пакету. Метою таких дій є приховання справжнього відправника пакету або знешкодження інформаційної системи шляхом здійснення DoS-атаки, оскільки серверу важче оброблювати багато запитів, кожен з яких надходить наче з різних адрес.

Для захисту від такої атаки можна використовувати фільтрацію пакетів на мережевому екрані, що заборонить пересилання небажаних пакетів у локальну мережу, також допомагає протидіяти цьому виду атаки обраний протокол TCP, оскільки в його пакетах наявні номери послідовності, і зловмиснику доведеться вгадувати ці номери задля перехоплення з’єднання.

2.5 Висновки за розділом

В ході розділу було розглянуто питання організації роботи розроблюваного комплексу засобів захищеного обміну повідомленнями. Розглянуто інструменти шифрування та хешування, що знадобляться у ході розробки. Освітлено деякі питання безпечності такого підходу до реалізації.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМПЛЕКСУ ЗАСОБІВ ЗАХИЩЕНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

3.1 Вибір інструментів для розробки

Для розробки серверної і клієнтської частин додатку обрано редактор коду Microsoft Visual Studio Code[10]. Цьому слугували наступні причини:

- Мінімалістичний та зручний інтерфейс редактору
- Наявність системи IntelliSense, яка допомагає у процесі написання коду підсвічуванням синтаксису мови програмування, помилок, за їх наявності, та рекомендаціями щодо оптимізації розроблюваного засобу.

Мовою програмування для розробки засобу обрано Python[11], оскільки це кросплатформенна мова, що дозволяє розробленому додатку працювати не лише на операційній системі Microsoft Windows, але й на різноманітних дистрибутивах Linux, MacOS та навіть Android.

В якості системи управління базою даних вирішено обрати СУБД sqlite3, тому що для неї у Python існує зручна бібліотека sqlite3[12], яка значно спрощує роботу програми із базою даних і має вбудовані інструменти для одночасної роботи з базою даних декількох суб’єктів.

3.2 Розробка програмного забезпечення клієнтської частини

Програмний код клієнтської частини знаходиться у додатку Б.

Робота додатку передбачає наступний ряд наявних функцій:

- Ідентифікація та аутентифікація абонента
- Написання текстових повідомлень
- Зашифрування і розшифрування тексту повідомлення симетричним криптоалгоритмом
- Передача та прийом зашифрованих повідомлень

- Виведення тексту відправлених та отриманих повідомлень у вікні клієнта
- Оновлення історії листування із іншим клієнтом при надходженні сигналу про нове повідомлення
- Відображення наявних у мережі клієнтів

Інтерфейс клієнтського додатку наведено на рисунку 3.1.

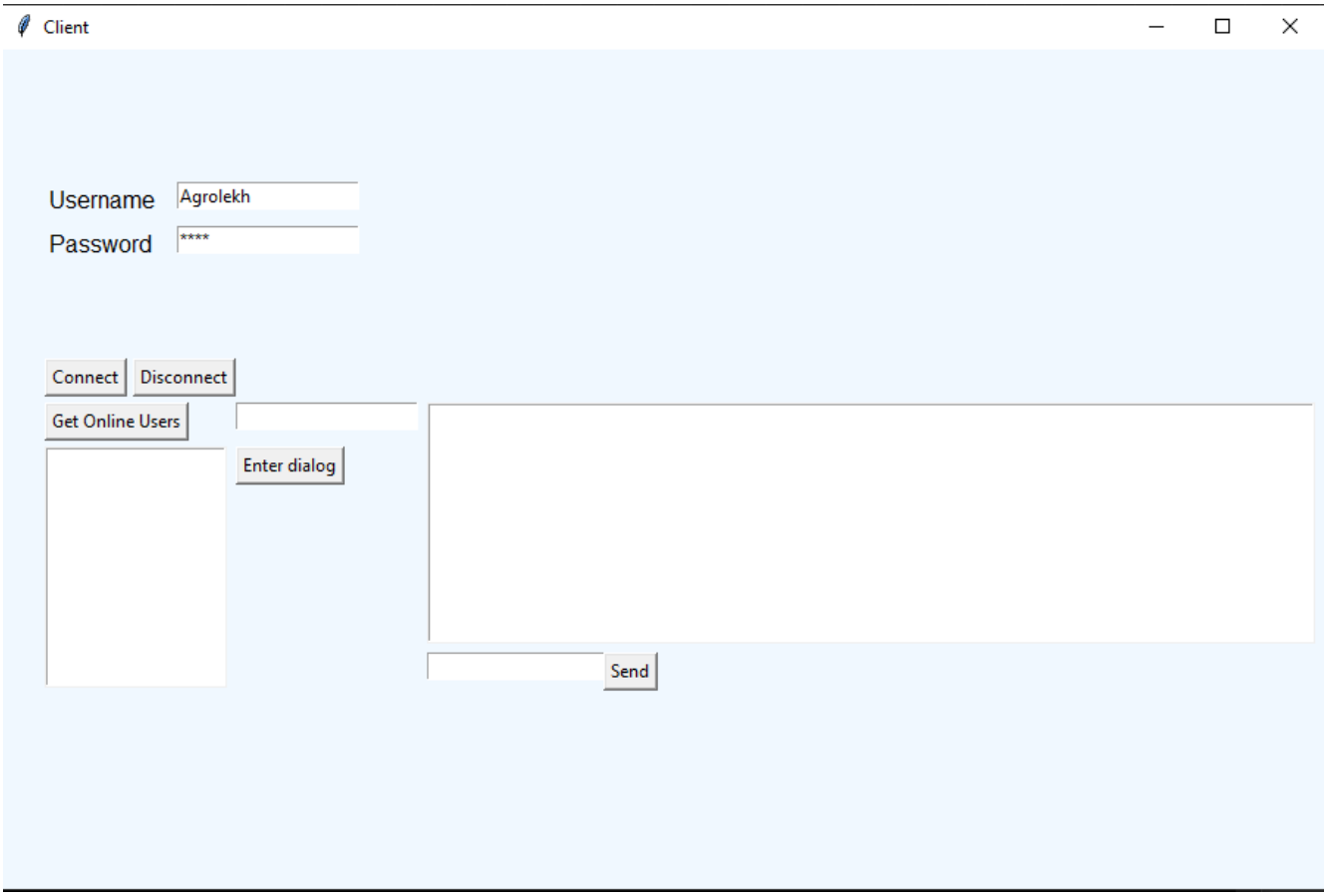

Рисунок 3.1. – Інтерфейс клієнтської частини додатку

Абонент аутентифікується у додаток за допомогою виданих йому адміністратором серверу логіну та паролю, які йому потрібно ввести у відповідні поля у вікні додатку. Введені логін та пароль відправляються на сервер на перевірку. Якщо дані введені користувачем вірні, то йому

дозволяється вхід у додаток для подальшого спілкування із іншими користувачами. Схему аутентифікації наведено на рисунку 3.2.

Блок 1 – початок підпрограми аутентифікації

Блок 2 – введення паролю користувача у відповідний рядок вводу даних

Блок 3 – обчислення хеш-функції від введеного паролю

Блок 4 – порівняння обчисленої хеш-функції із хеш-функцією обчисленою при реєстрації користувача, що зберігається у базі даних сервера

Блок 5 – повідомлення користувача про помилку у введеному паролі і повернення до етапу введення пароля

Блок 6 – підпрограма надання користувачу доступу до системи

Блок 7 – кінець підпрограми аутентифікації

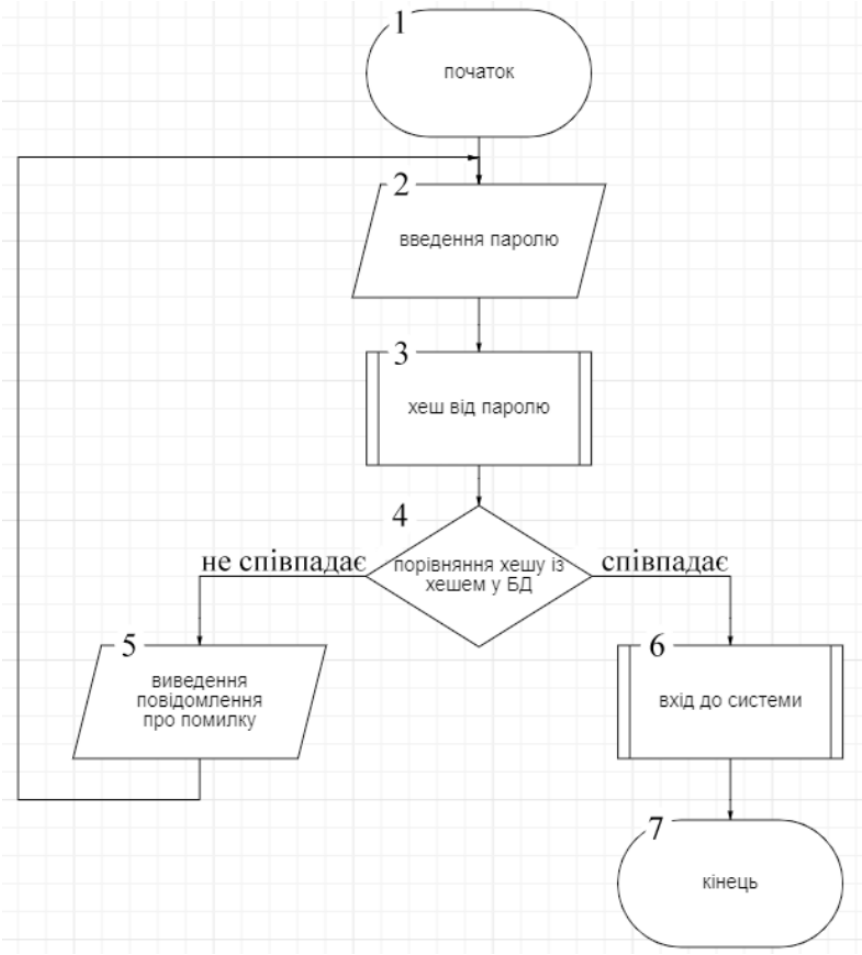

Рисунок 3.2. Схема аутентифікації користувача у додатку

Процес хешування реалізовано завдяки відкритій бібліотеці, написаній мовою Python, hashlib[13], яка надає дуже зручний функціонал у використанні багатьох функцій хешування.

Для відправлення та отримання повідомлень незахищеним каналом зв’язку клієнтська програма зашифровує повідомлення у випадку відправлення даних і розшифровує їх у випадку отримання даних. Шифрування повідомлень здійснюється за алгоритмом AES (Advanced Encryption Standard)[14] із ключем довжиною 256 біт. Після того, як повідомлення було зашифровано, його треба піддати транспортному кодуванню для того, щоб не втратити службових символів і інших даних у процесі їх передачі, збереження до бази даних та отримання із бази даних. Після отримання повідомлення його піддають оборотному процесу від транспортного кодування, щоб отримати його у первинному вигляді, і подальшому дешифруванню.

Процеси шифрування і дешифрування реалізовані завдяки відкритій бібліотеці, написаній мовою Python, pyaes[15], яка надає гнучкий та зручний функціонал у виборі режиму роботи криптоалгоритма та подальшим діям із шифрування даних.

Для того, щоб програма-клієнт мала змогу одночасно обробляти команди подані користувачем через графічний інтерфейс та команди сервера, передбачено обробку команд з різних джерел у різних потоках: тобто команди користувача виконуються у основного потоці, а у допоміжному потоці відбувається прослуховування нових повідомлень, в яких знаходяться службові команди, що надходять до клієнта із сервера. Приблизну схему потокової взаємодії наведено на рисунку 3.3.

Рисунок 3.3. – Схема багатопотокової роботи клієнта

3.3 Розробка програмного забезпечення серверної частини

Програмний код серверної частини знаходиться у додатку А.

Функціонал серверу передбачає організація і підтримання TCP-з’єднання між клієнтами, додавання нових користувачів у систему, за необхідності. Інтерфейс серверної частини додатка наведено на рисунку 3.4.

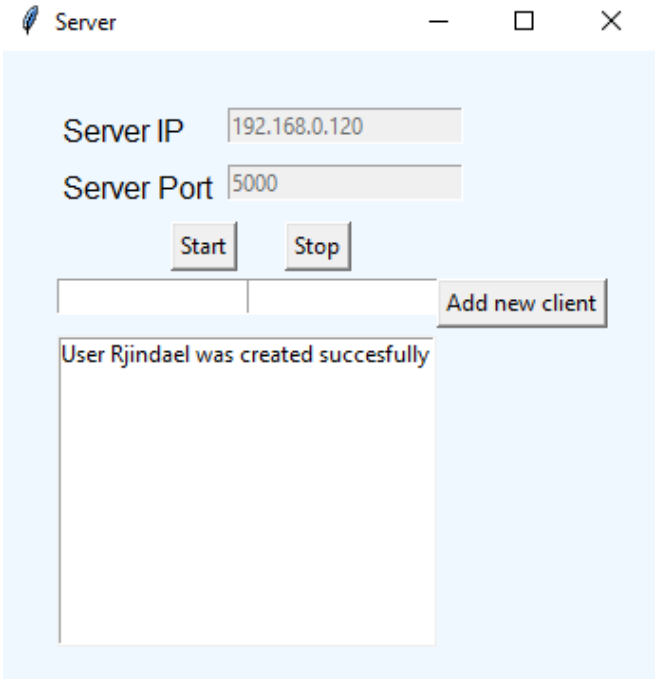

Рисунок 3.4 – Інтерфейс серверної частини додатку

Для запуску сервера необхідно налаштувати адресу та порт, на якій він буде працювати та приймати запити від клієнтів.

Сервером передбачена можливість додавання нових користувачів у систему шляхом введення їх майбутніх логіну та паролю. Після натискання кнопки «Add user» із паролю майбутнього користувача береться хеш-функція за алгоритмом SHA-256 та створюється запис у базу даних із логіном та хешем паролю користувача. Із цим записом і будуть порівнюватись введені користувачем під час аутентифікації дані.

Аналогічно ситуації із клієнтом, серверу потрібно декілька потоків для роботи, а саме 2+N потоків, де N – кількість одночасно під’єднаних до сервера клієнтів. Основний потік виводить графічний інтерфейс і приймає та оброблює команди адміністратора, допоміжний потік приймає та налаштовує TCP-з’єднання із клієнтами та ще N потоків займаються обробкою команд, що надходять від конкретних під’єднаних і аутентифікованих користувачів. Приблизну схему багатопокової роботи сервера наведено на рисунку 3.5.

Рисунок 3.5. Схема багатопокової роботи сервера

3.4 Висновки за розділом

В ході виконання розділу було обрано інструментальні засоби розробки програмного забезпечення комплексу, що складається із клієнтської та серверної частин. Розроблено алгоритми та програмне забезпечення комплексу. Комплекс дозволяє організувати процес захищеного обміну між користувачами.

4 ДОСЛІДЖЕННЯ ЧАСОВИХ ХАРАКТЕРИСТИК РОБОТИ ЗАХИСНИХ МЕХАНІЗМІВ РОЗРОБЛЕНОГО КОМПЛЕКСУ ЗАСОБІВ

4.1 Постановка задачі

У даному розділі пропонується розглянути процеси шифрування та дешифрування повідомлень з позиції їх швидкодії (мається на увазі час виконання певної операції) на різних електронно-обчислювальних машинах. Планується використати декілька різних комп'ютерів та список із повідомлень різної довжини, які будуть шифруватись двома різними криптоалгоритмами.

Оскільки розглядувані криптоалгоритми симетричні, то вважатимемо, що час затрачений на розшифрування буде рівним або порівняним із часом, затраченим на зашифрування.

У порівнянні примуть участь такі криптоалгоритми:

- AES-256
- RC4

У реалізації алгоритма RC4 використаємо відкриту бібліотку rc4[16].

Для участі у дослідженні обрано два комп'ютери різної конфігурації:

1) Процесор: Intel Core i5-10600
   Оперативна пам'ять: DDR4 16 GB 2666 МГц
   Частота роботи процесору: 4.1 ГГц
   Операційна система: Windows 10 Enterprise N version 20H2(x64)
2) Процесор: Intel Core i3-6006
   Оперативна пам'ять: DDR3 8 GB 1333 МГц
   Частота роботи процесору: 2.0 ГГц
   Операційна система: Windows 10 Enterprise N version 20H2(x64)

Кількістю ядер та потоків процесору можна знехтувати, оскільки програмні реалізації бібліотек шифрування, що використані у комплексі

засобів, не передбачають багатопотокової роботи, тому збільшення кількості ядер не принесе жодного прискорення процесу шифрування.

Дослідження проводитиметься на довжинах повідомлень від 2^0 байт до 2^{21} байт. Хоча і відправка повідомлень настільки великої довжини малоймовірна, у цілях дослідження цікаво роздивитись більш повну картину залежності швидкості виконання криптографічних операцій від обраного алгоритму і наявного апаратного забезпечення.

Для дослідження підготовлено масив значень X, який містить у собі довжини, до яких буде збільшене повідомлення і надалі зашифроване:
X = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152]

Літерою Y позначимо масив значень часу, що був витрачений на зашифрування конкретного повідомлення. По цих двух масивах побудуємо на системі координат графіки і порівняємо швидкість роботи алгоритмів.

Проведемо по п’ять замірів часу для кожного етапу експерименту і використаємо усереднені дані з цих замірів.

4.2 Дослідження часових характеристик шифрування повідомлень криптоалгоритмом AES-256

Реалізацію алгоритму AES-256 використовуватимемо ту саму, що й при розробці програмного засобу обміну захищеними повідомленнями.

По закінченні роботи програми отримано наступний масив значень часу (у секундах) Y:
Y = [0.000158799, 0.000133199, 0.0001267, 0.0001197, 0.000121099, 0.000156299, 0.0002374, 0.0004175, 0.0007301, 0.001420899, 0.002677499, 0.0052993, 0.010760499, 0.020930899, 0.0434449, 0.0861969, 0.1775847, 0.331943899, 0.693148199, 1.343429899, 2.7260595, 5.3780279]

За результатами роботи алгоритму побудовано графік і приведено на рисунку 4.1.

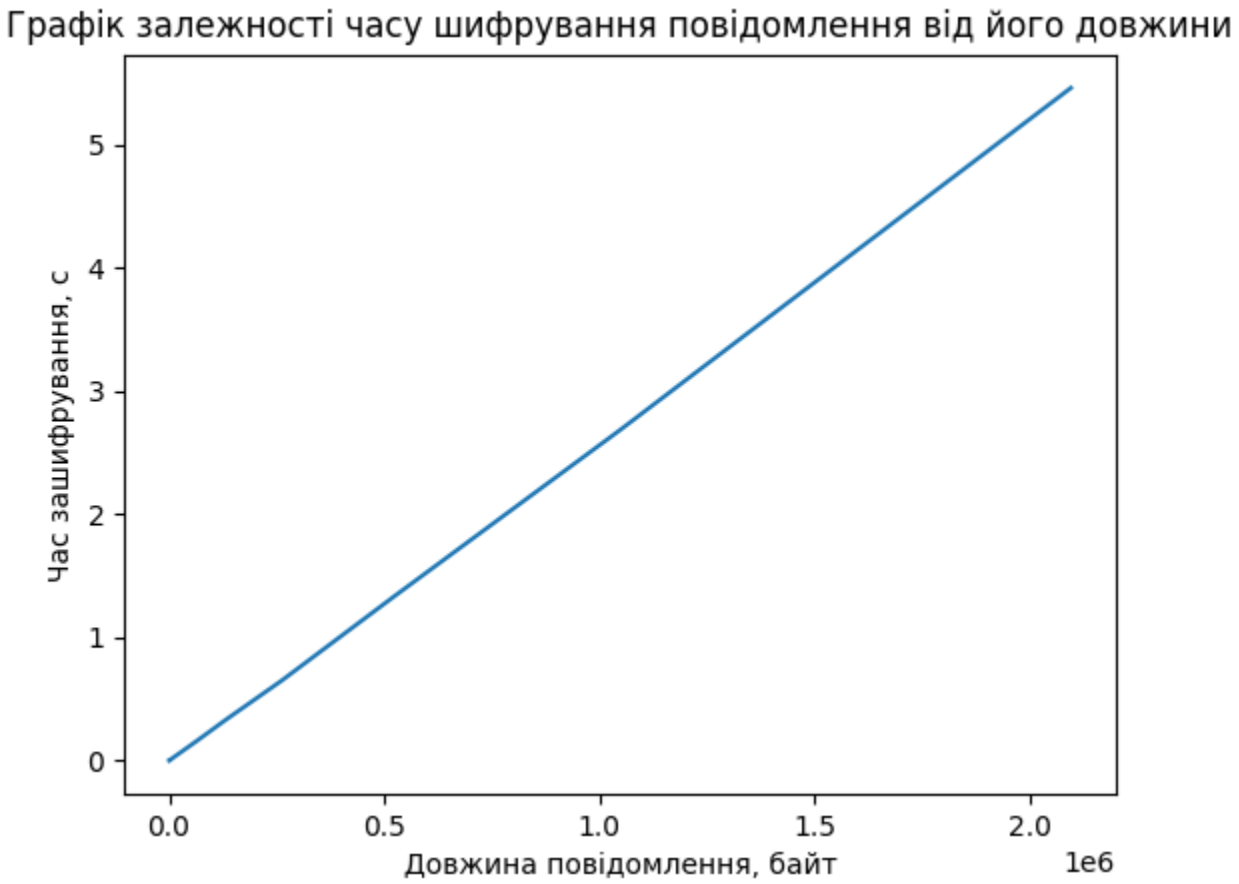

Рисунок 4.1. – Графік залежності часу зашифрування повідомлення від його довжини для комп’ютера із процесором i5-10600 у випадку AES-256

Для менш потужного комп’ютера у випадку AES-256 отримуємо наступний масив значень часу Y:

Y = [0.000368899, 0.0004822, 0.000510899, 0.0003326, 0.000388699, 0.000424699, 0.000949, 0.0014552, 0.003645899, 0.0039283, 0.007794999, 0.018858199, 0.040494099, 0.107768099, 0.134566499, 0.288856099, 0.625579099, 1.227337399, 2.3977218, 5.010818, 9.761195299, 19.96970069]

За результатами роботи алгоритму побудовано графік і приведено на рисунку 4.2.

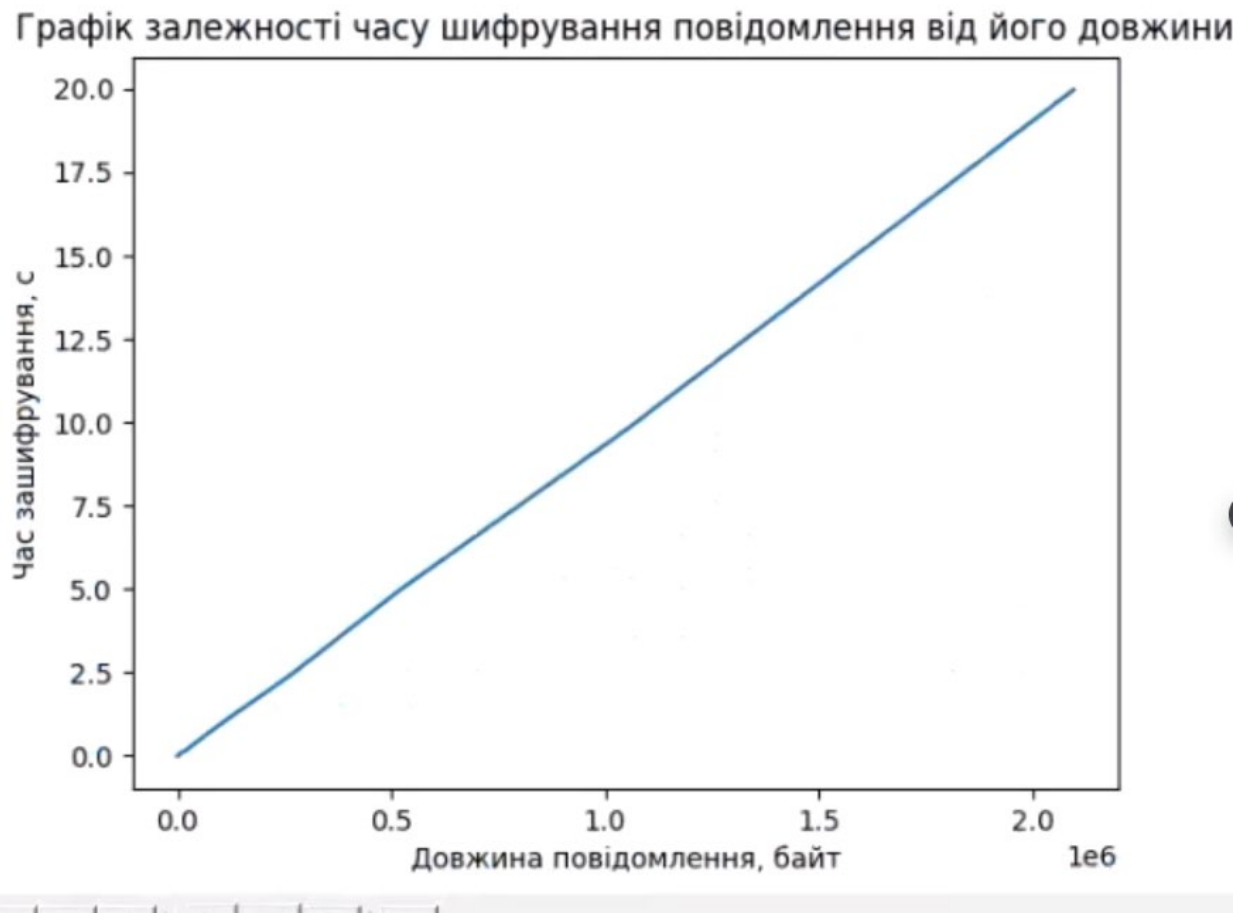

Рисунок 4.2. – Графік залежності часу зашифрування повідомлення від його довжини для комп’ютера із процесором i3-6006 у випадку AES-256

Для більш наглядного порівняння роботи двох різних комп’ютерів, розглянемо ці два графіки залежності часу шифрування від довжини в одній системі координат. Порівняльний графік приведено на рисунку 4.3.

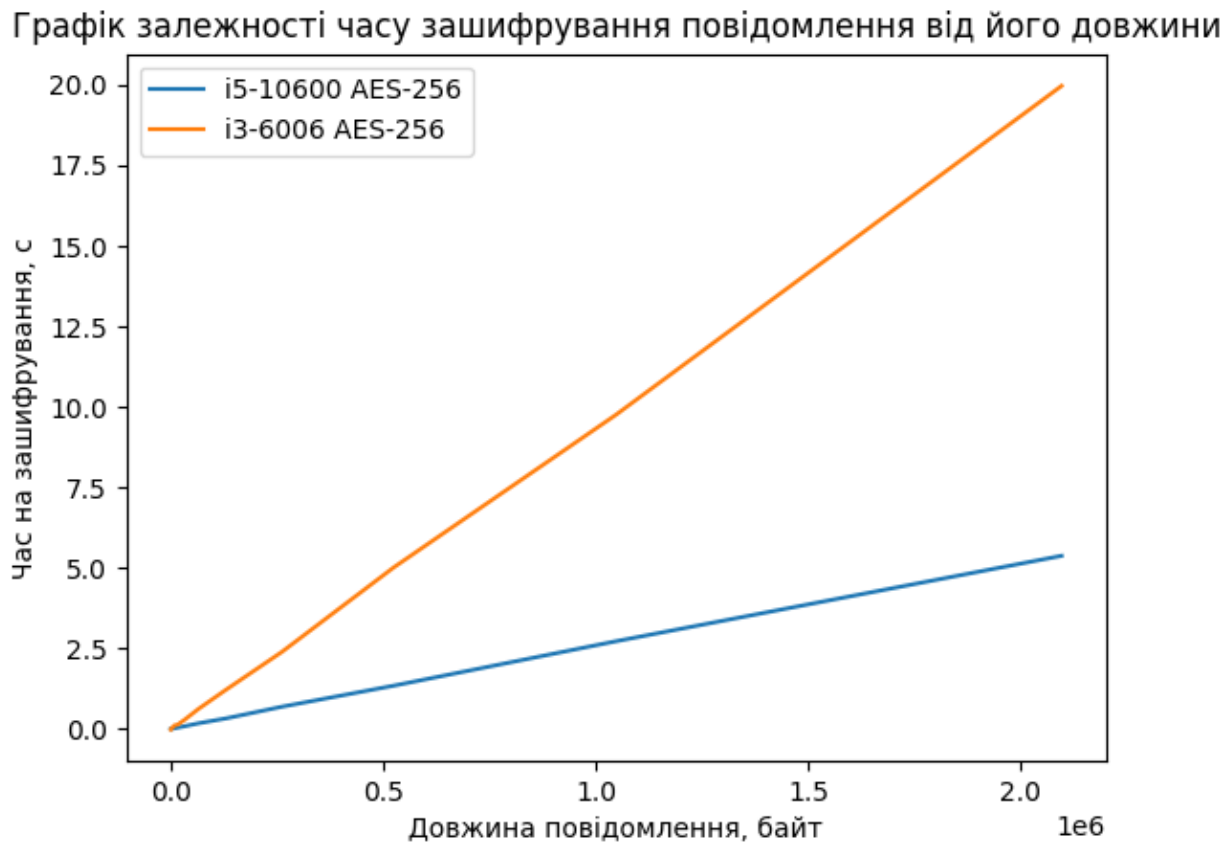

Рисунок 4.3. – Сумісний графік часу шифрування на двох комп'ютерах для AES-256

4.3 Дослідження часових характеристик шифрування повідомлень криптоалгоритмом RC4

Криптоалгоритм RC4, на відміну від AES-256, є потоковим шифром. Потокові шифри зазвичай швидші за блокові і простіші у програмній реалізації. Вони частіше використовуються у апаратних рішеннях, ніж у програмних.

Для алгоритму RC4 використовуємо ті самі значення довжин повідомлення і по завершенні роботи програми отримуємо наступні значення масиву Y:

Y= [0.0000963, 0.0000758, 0.0000732, 0.0000855, 0.000086599, 0.0000966, 0.0000927, 0.0001195, 0.0001529, 0.000229699, 0.000435199, 0.000708999,

0.0013109, 0.0025251, 0.0051088, 0.009738099, 0.0191068, 0.0393533, 0.079026699, 0.158065099, 0.3215449, 0.6378145]

За результатами роботи алгоритму побудовано графік і приведено на рисунку 4.4.

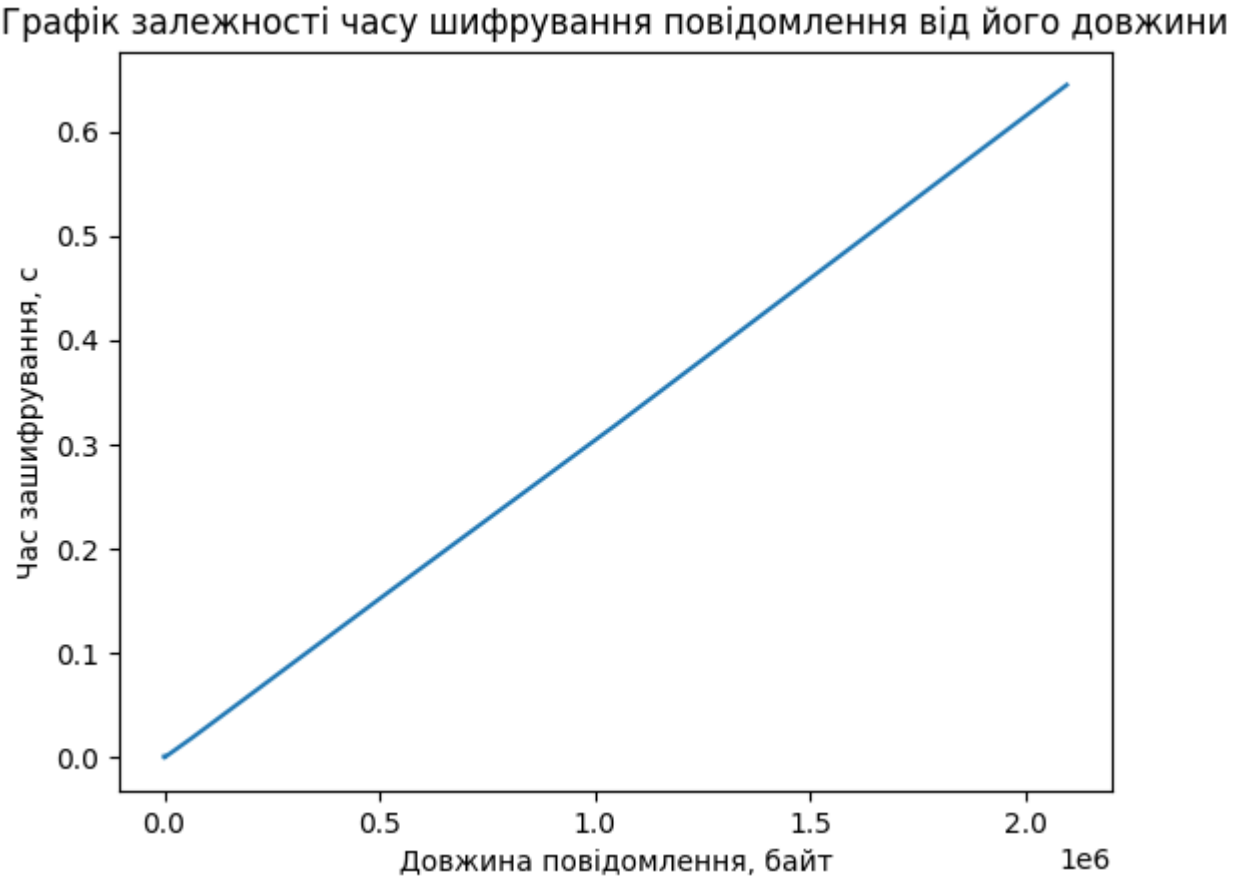

Рисунок 4.4 - Графік залежності часу зашифрування повідомлення від його довжини для комп'ютера із процесором i5-10600 у випадку RC4

І для менш потужного комп'ютера отримуємо такі величини затраченого часу:

Y = [0.0002276, 0.0003767, 0.000355699, 0.000347399, 0.0003561, 0.0003824, 0.0004311, 0.000790299, 0.0008193, 0.0022509, 0.002513899, 0.0036694, 0.0172265, 0.022923799, 0.035544399, 0.0879216, 0.157485599, 0.1895504, 0.348844999, 0.7197541, 1.6961928, 3.8082066]

За результатами роботи алгоритму побудовано графік і приведено на рисунку 4.5.

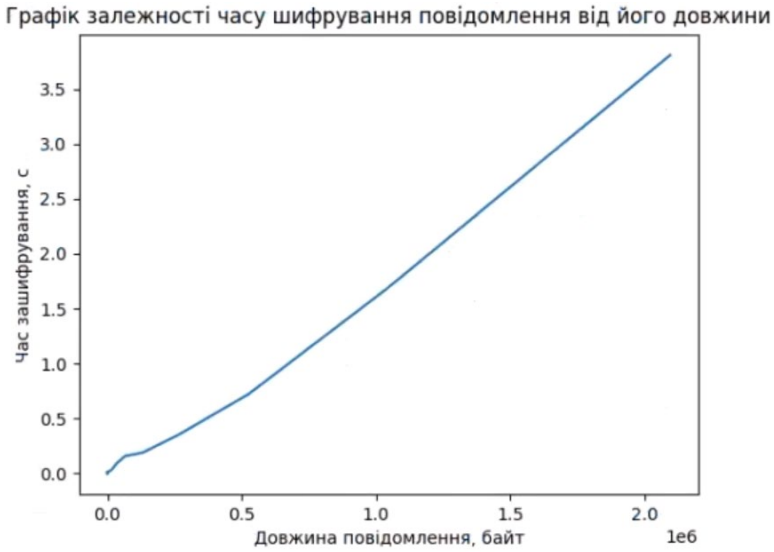

Рисунок 4.5. Графік залежності часу зашифрування повідомлення від його довжини для комп’ютера із процесором i3-6006 у випадку RC4

Для більшої наглядності різниці у швидкості роботи алгоритмів на комп’ютерах різної потужності графік приведено на рисунку 4.6.

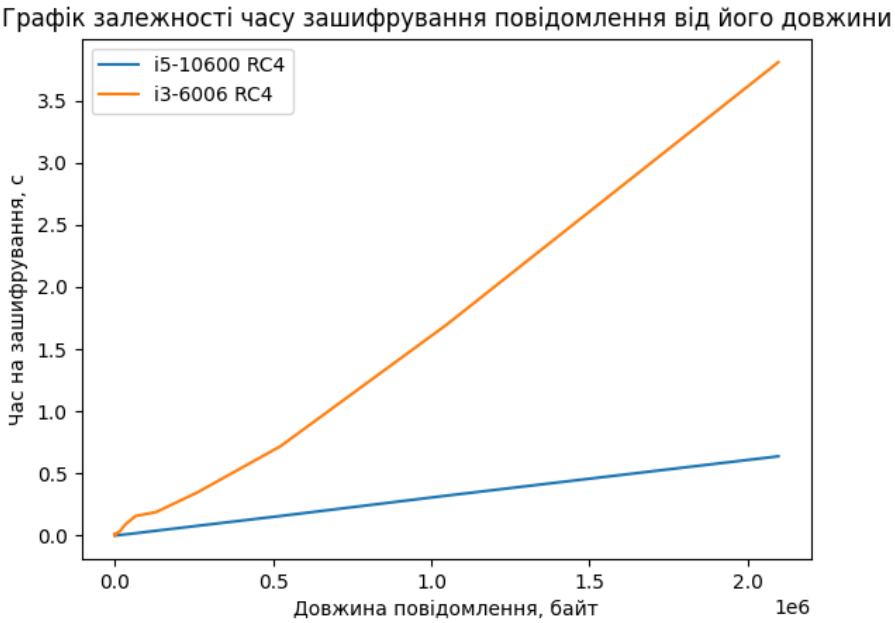

Рисунок 4.6 - Сумісний графік часу шифрування на двох комп’ютерах для RC4

4.4 Узагальнення та аналіз отриманих даних

Дані, отримані у ході дослідження, приведено у таблиці 4.1.

Таблиця 4.1 – Залежність часу виконання криптографічної операції від довжини шифрованого повідомлення

Довжина повідомлення, байт	Час шифрування повідомлення, секунд			
	AES-256		RC4	
	i5-10600	i3-6006	i5-10600	i3-6006
1	0.000158799	0.000368899	0.0000963	0.0002276
2	0.000133199	0.0004822	0.0000758	0.0003767
4	0.0001267	0.000510899	0.0000732	0.000355699
8	0.0001197	0.0003326	0.0000855	0.000347399
16	0.000121099	0.000388699	0.000086599	0.0003561
32	0.000156299	0.000424699	0.0000966	0.0003824
64	0.0002374	0.000949	0.0000927	0.0004311
128	0.0004175	0.0014552	0.0001195	0.000790299
256	0.0007301	0.003645899	0.0001529	0.0008193
512	0.001420899	0.0039283	0.000229699	0.0022509
1024	0.002677499	0.007794999	0.000435199	0.002513899
2048	0.0052993	0.018858199	0.000708999	0.0036694
4096	0.010760499	0.040494099	0.0013109	0.0172265
8192	0.020930899	0.107768099	0.0025251	0.022923799
16384	0.0434449	0.134566499	0.0051088	0.035544399
32768	0.0861969	0.288856099	0.009738099	0.0879216
65536	0.1775847	0.625579099	0.0191068	0.157485599
131072	0.331943899	1.227337399	0.0393533	0.1895504
262144	0.693148199	2.3977218	0.079026699	0.348844999
524288	1.343429899	5.010818	0.158065099	0.7197541
1048576	2.7260595	9.761195299	0.3215449	1.6961928
2097152	5.3780279	19.96970069	0.6378145	3.8082066

На рисунку 4.7. приведено графік порівняння усіх результатів за розділом.

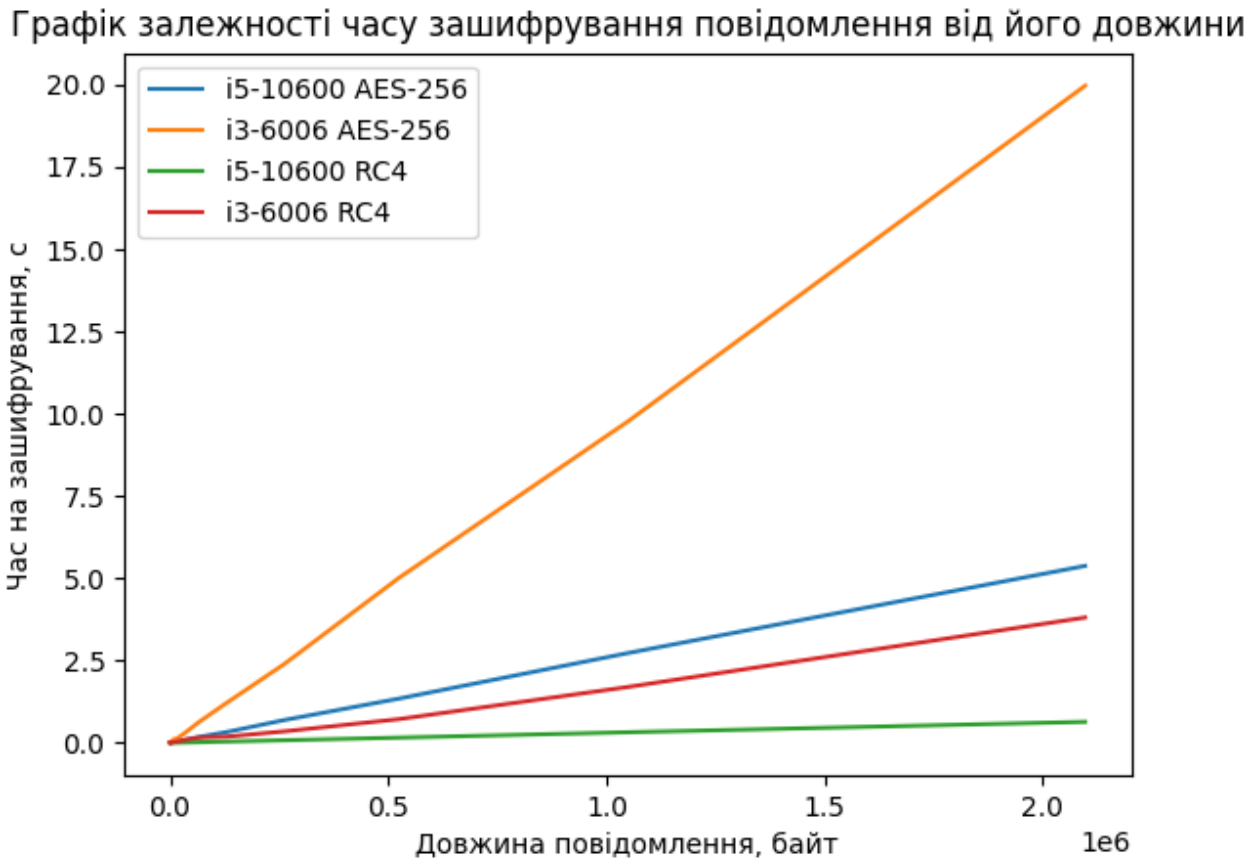

Рисунок 4.7 – Залежність часу шифрування від довжини повідомлення для двох комп'ютерів та двох криптоалгоритмів

Графік явно показує, наскільки сильно зростає час виконання криптографічних операцій для блокових шифрів, на відміну від потокових шифрів. З цього можна зробити висновок, що у випадку, коли наявне апаратне забезпечення не надає великої обчислювальної потужності і довжина передаваних повідомлень дуже значна, то може бути доцільним використовувати потокові шифри замість блокових. Але автор вважає, що такий сценарій не є настільки розповсюдженим, тому у загальному випадку цією різницею у часі виконання операції шифрування можна знехтувати.

За думкою автора, корисним буде розглянути початкові точки графіку у диапазоні довжин повідомлення від 1 до 1024 байт, оскільки рідко коли люди відправляють повідомлення у месенджерах довже 700 символів.

Приведемо і порівняємо дані у вигляді графіка на рисунку 4.8.

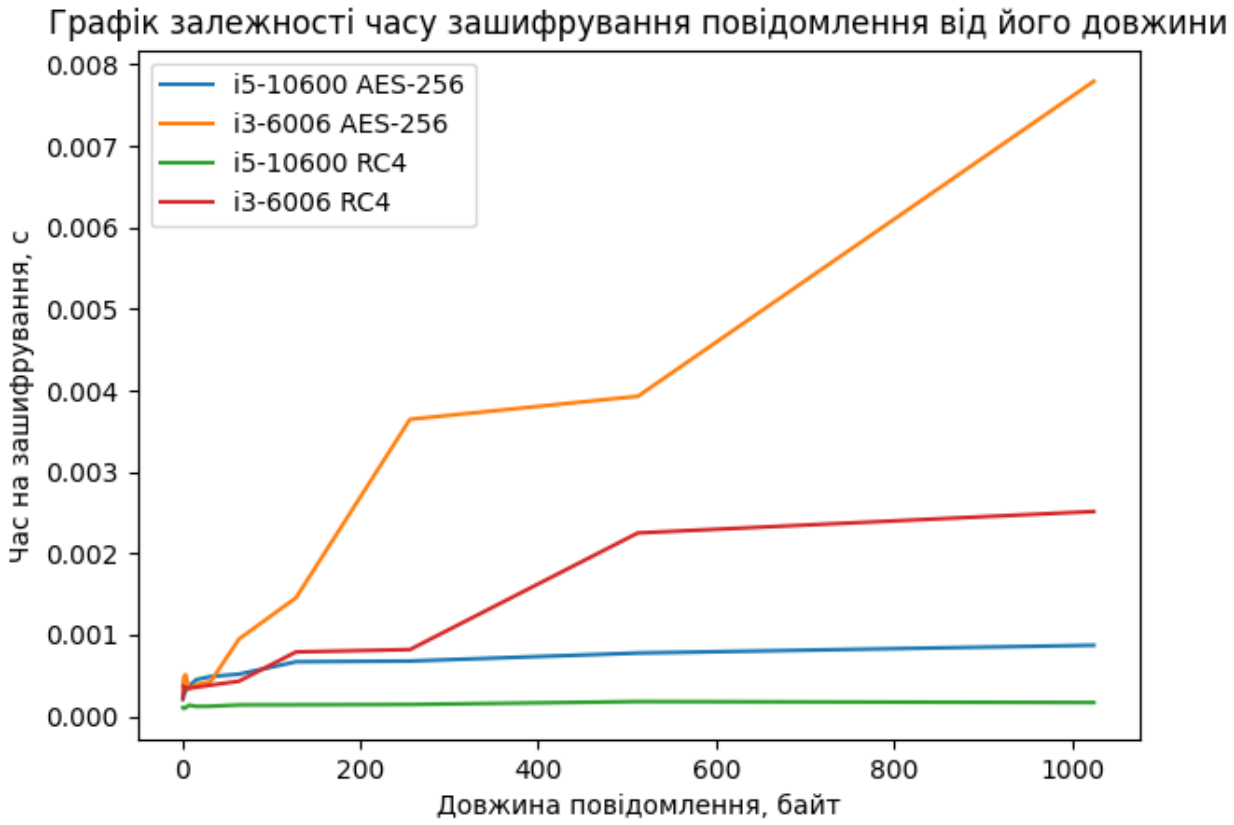

Рисунок 4.8 – Графік залежності часу шифрування повідомлення від його довжини для випадку відправлення коротких повідомлень

Як бачимо із графіка на коротких повідомленнях криптоалгоритм RC4 показує більш швидкий результат, але для загального випадку листування між двома людьми ця різниця буде незначною, бо складає усього лиш соті долі секунди. Криптоалгоритм AES надає достатню швидкість, яка є прийнятною для засобів обміну повідомленнями, і при цьому є стійкішим за криптоалгоритм RC4, тому у розробленому засобів доцільно використовувати саме його.

4.5 Висновки за розділом

В ході розділу було проведено дослідження швидкості роботи функцій шифрування на прикладі двох машин: одна із процесором Intel Core i5-10600,

друга – із процесором Intel Core i3-6006, та на прикладі двох криптоалгоритмів: RC4 та AES-256. Експеримент показав, що недоцільно використовувати потокові криптоалгоритми для задачі шифрування коротких повідомлень, адже виграш у часі у порівянні із блоковим криптоалгоритмом незначний і варто віддати перевагу надійності блокових криптоалгоритмів.

5 МЕТОДИКА ВИКОРИСТАННЯ РОЗРОБЛЕНОГО КОМПЛЕКСУ ЗАСОБІВ

5.1 Інструкція з використання комплексу

Для початку використання засобу, адміністратор серверу повинен запустити сам сервер, обравши для трансляції з’єднання по мережі адресу та порт. Зробити це можна шляхом введення цих даних у відповідні поля у вікні додатку та натиском кнопки «Start», як зображено на рисунку 5.1.

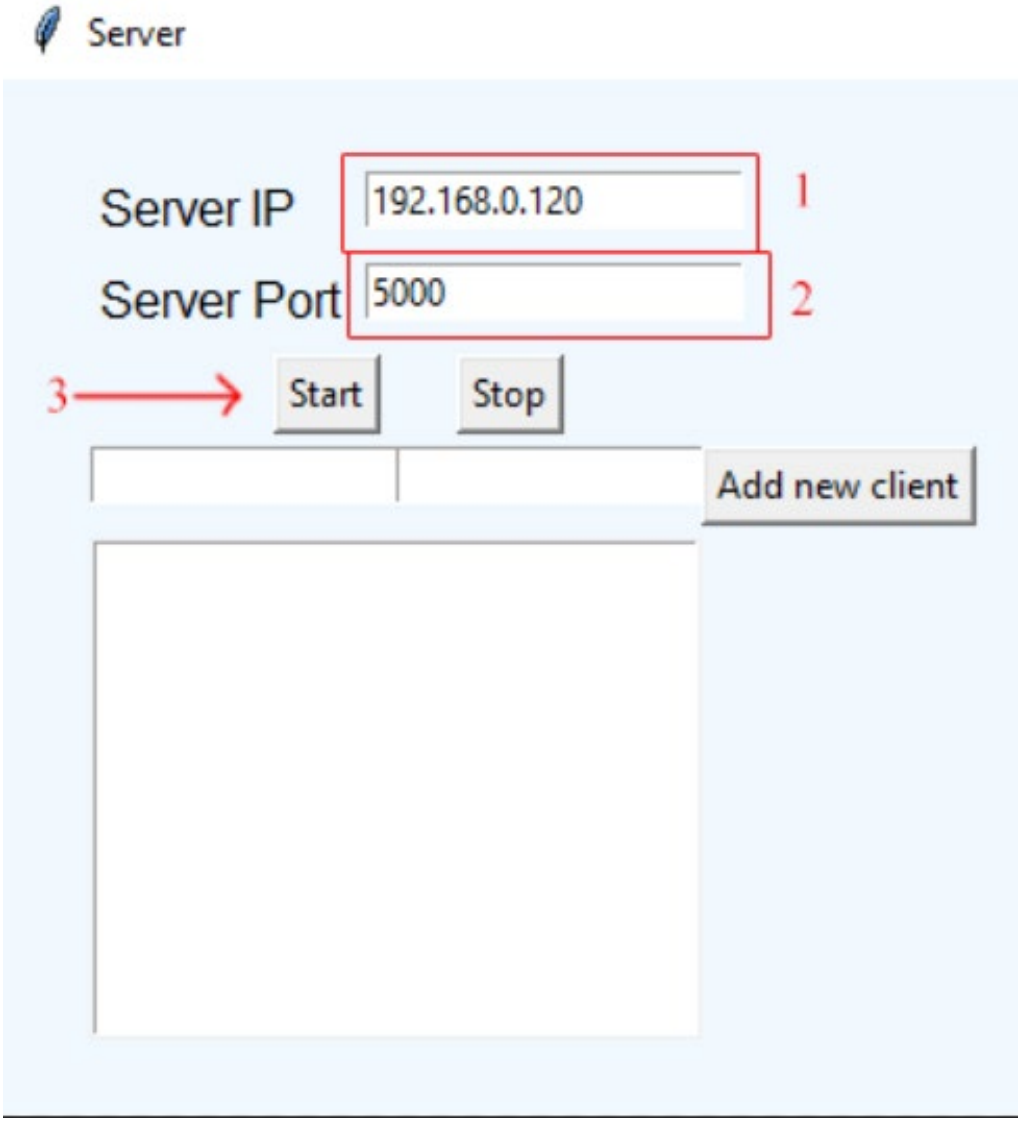

Рисунок 5.1. – Процедура запуску сервера

Після того як сервер було запущено, можна створити облікові записи користувачів, якщо їх ще не існує. Зробити це можна введенням даних користувачів у поля вводу даних поруч із кнопкою «Add new client», як показано на рисунку 5.2.

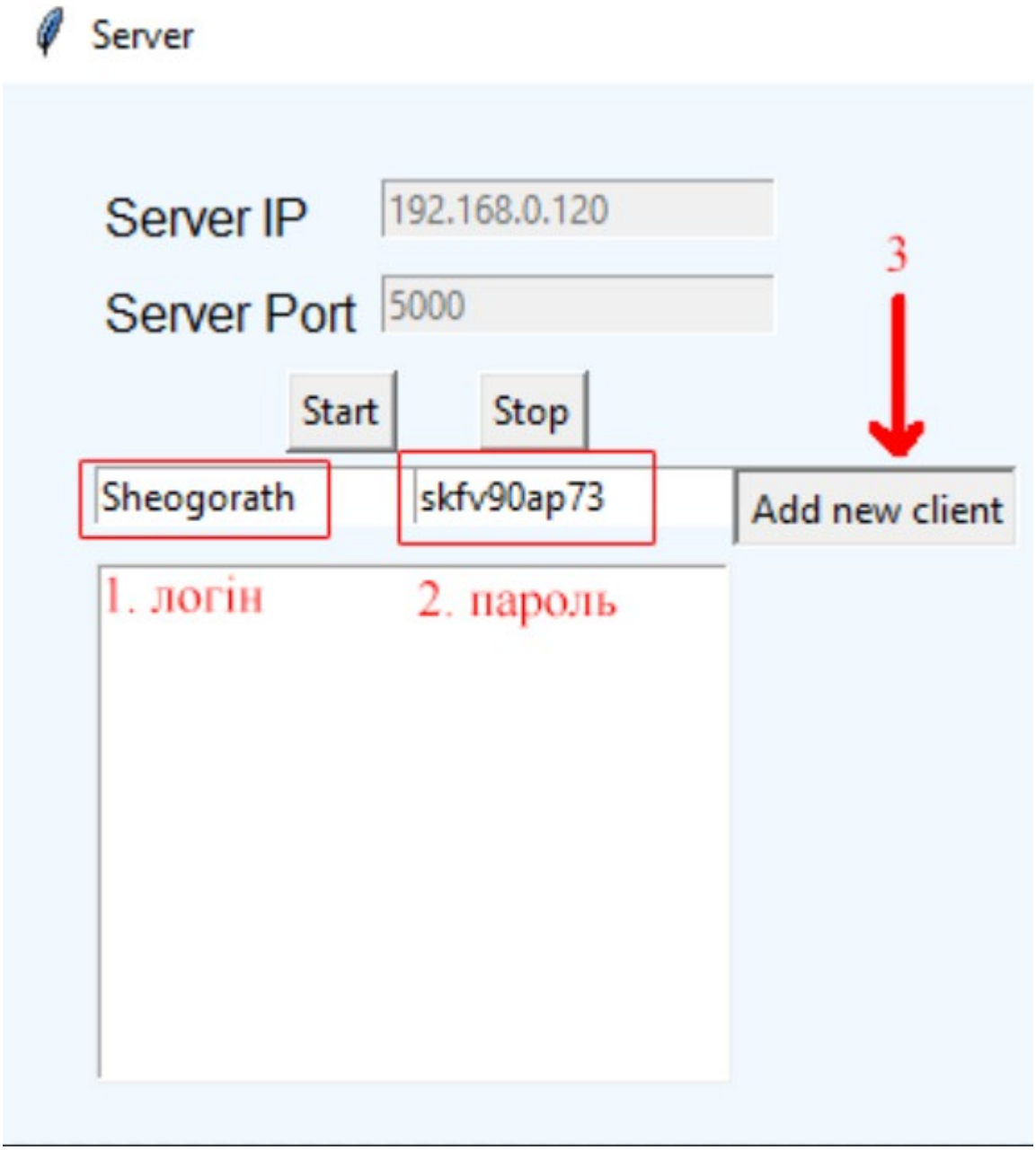

Рисунок 5.2. – Процедура створення нового користувача

Після створення нового користувача, відобразиться повідомлення про успіх виконання операції або повідомлення про помилку, у випадку проблем із базою даних або попередньої наявності користувача із таким логіном.

Після того, як було створено щонайменше два користувача, процес спілкування між ними може бути розпочато. Для цього користувачеві потрібно включити свою програму-клієнт та підключитись до адреси та порту серверу, аутентифікувавшись за допомогою свого логіну та паролю. Цей процес зображено на рисунку 5.3.

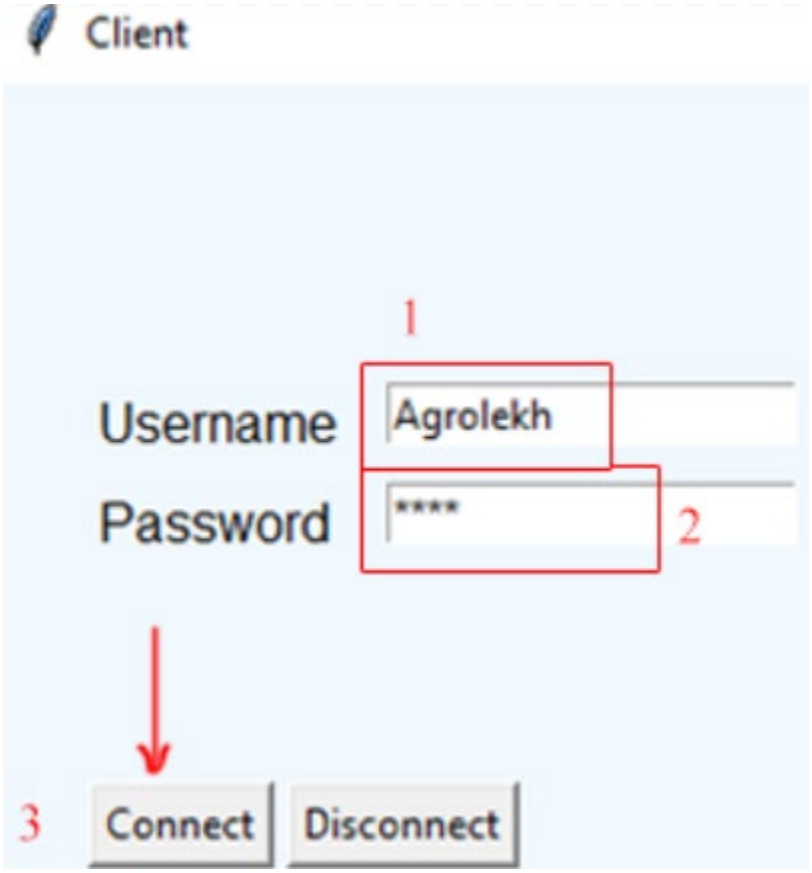

Рисунок 5.3. Процес аутентифікації користувача

Конфігурація ж адреси та порту сервера прихована з інтерфейсу і вводиться у конфігураційний файл, з якого зчитуються ці дані після натискання кнопки «Connect». Структуру конфігураційного файлу наведено на рисунку 5.4.

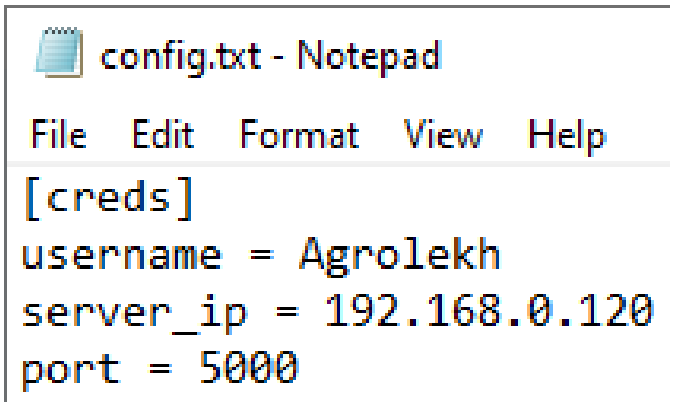

Рисунок 5.4. – Конфігураційний файл клієнта

Крім адреси та порту сервера, у конфігураційному файлі запам’ятовується останній використаний користувачем логін для зручності використання.

Якщо користувач із таким логіном та паролем зарєєстрований у системі, то результат аутентифікації успішний і можна приступити до процесу спілкування. Перед цим можна отримати список користувачів, які користуються системою прямо зараз. Відбувається це натиском на кнопку «Get Online Users», яка надсилає запит до сервера і повертає список користувачів онлайн. Цей процес відображено на рисунку 5.5.

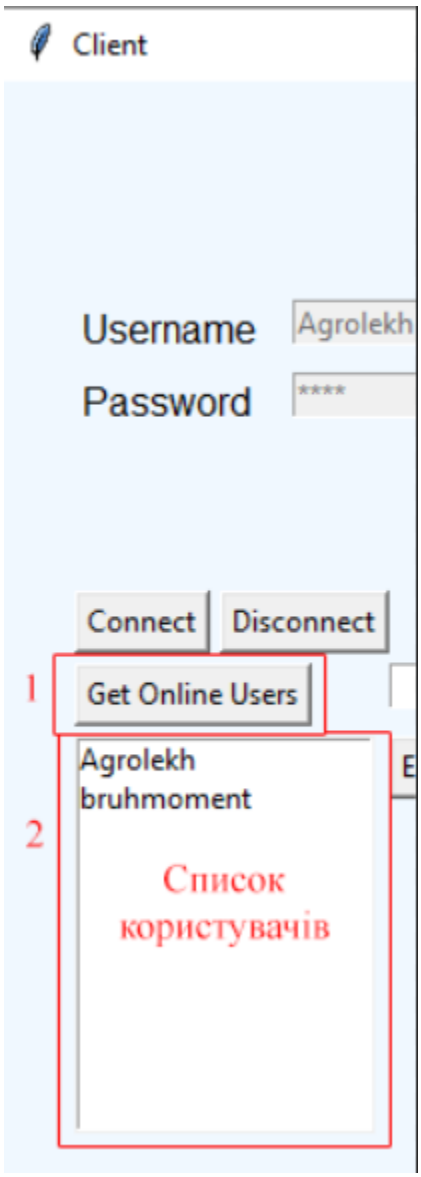

Рисунок 5.5. – Процес отримання списку користувачів онлайн

Для початку розмови із кимось треба ввести його логін у поле вводу даних поруч із кнопкою «Enter dialog», як зображено на рисунку 5.6.

Якщо це перший діалог із користувачем, то обидва користувачі повинні знаходитись у мережі для того, щоб згенерувати спільний ключ симетричного шифрування. Якщо ж до цього діалог вже відбувався, ця умова необов’язкова, можна просто розпочати діалог і отримувач зможе ознайомитись із повідомленнями у свій наступний вхід до системи.

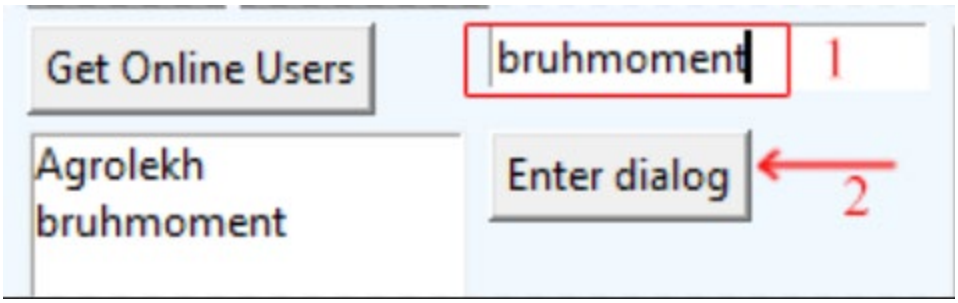

Рисунок 5.6. – Процедура входу у діалог із іншим користувачем

Результат входу до діалогу і відкритий чат із іншим користувачем зображено на рисунку 5.7.

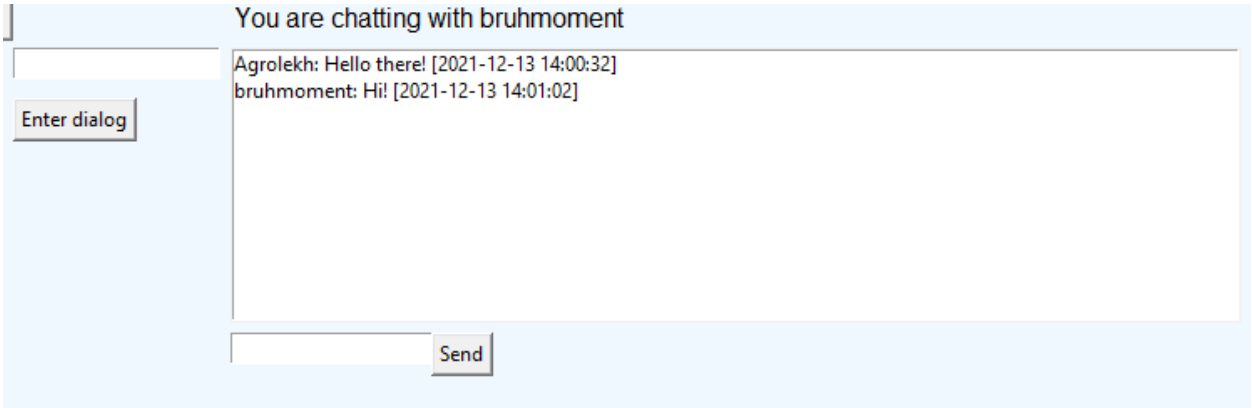

Рисунок 5.7. – Відкрите листування із іншим користувачем

Для відправки повідомлення його потрібно ввести у поле вводу поруч із кнопкою «Send» та натиснути кнопку «Send», як показано на рисунку 5.8.

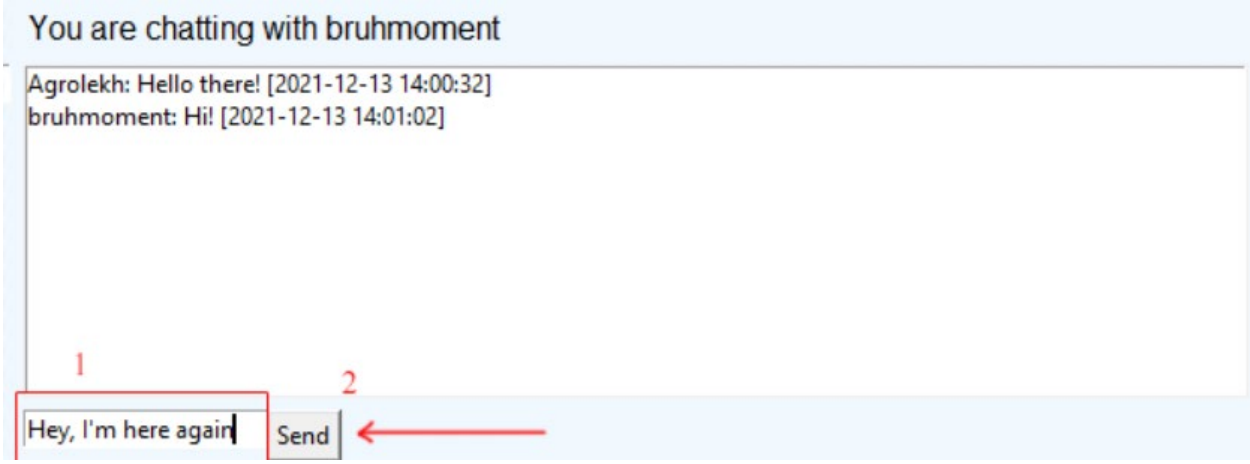

Рисунок 5.8. – Відправка повідомлення

Результат відправки повідомлення співрозмовнику бачимо на рисунку 5.9.

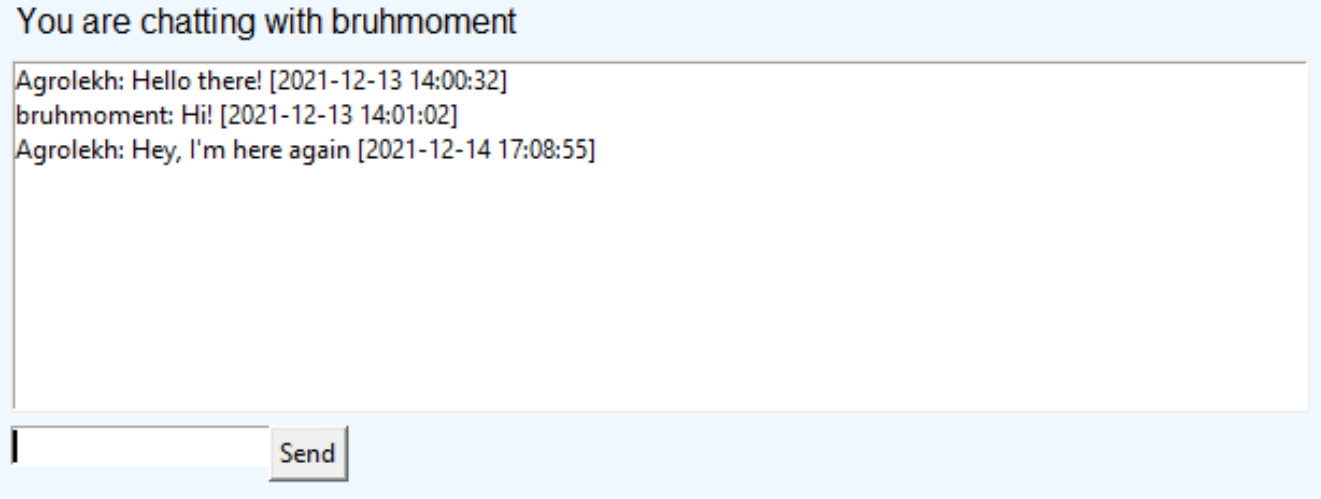

Рисунок 5.9. – Нове повідомлення, надіслане у чат

5.2 Можливості використання розроблених засобів у цілях навчання

Розроблений комплекс можна використовувати у навчальних цілях для ознайомлення студентів із базовими поняттями у області захищеного обміну повідомленнями. Роботу із ознайомлення необхідно виконувати по групах. Можливий варіант, у якому двоє із студентів виступають у ролі

співрозмовників, а третій – у ролі зловмисника, який намагається перехопити дані зашифрованих повідомлень чи ключ, що передається у зашифрованому вигляді, наприклад, за допомогою утиліти Wireshark. Після успішного перехоплення пакету, можна переконатись у тому, що відкритий текст повідомлення недоступний до прочитання завдяки шифруванню.

5.3 Висновки за розділом

У даному розділі наведено інструкцію із використання розробленого комплексу засобів захищеного обміну повідомленнями.

Також запропонований варіант використання засобу у навчальних цілях з метою ознайомлення студентів із базовими принципами захищеного обміну повідомленнями.

6. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Вимоги безпеки праці під час виконання робіт на робочому місці

Згідно законодавства України щодо охорони праці, кожен працівник має право на безпечні та комфортні умови праці, що досягається комплексом із соціально-економічних, організаційно-технічних та санітарно-гігієнічних заходів.

Розділ складено і описано згідно затверджених Міністерством соціальної політики України «Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» №207 від 14.02.2018 року та Закону України «Про охорону праці» згідно Постанови Верховної Ради України № 77-VIII від 28.11.2014 року[16]. Даний розділ та його положення покликані забезпечити необхідний рівень безпеки та комфорту працівників задля зниження ризиків уникнення чинників, що негативно впливають на життя та здоров'я фахівця під час виконання його службових обов'язків.

Відповідно до НПАОП 0.00-7.15-18 [18] встановлено наступні вимоги безпеки до робочого місця працівника з екранними пристроями:

1) Робоче місце має забезпечувати працівникові простір для руху та зміни робочого положення.
2) Усі види випромінення мають бути зведені до гранично допустимого рівня.
3) Організація робочого місця з екранними пристроями має забезпечувати відповідність усіх психофізіологічних вимог, антропологічних та ергономічних вимог зважаючи на специфіку виконуваних робіт.
4) Освітлення робочого місця має відповідати вимогам Державних санітарних правил і нормам роботи з візуальними дисплейними

терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98 [19]

5) Мікроклімат виробничого приміщення з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [20]
6) Робоча поверхня повинна мати достатній розмір для гнучкого розміщення екрана, клавіатури, документів та відповідного обладнання і мати поверхню з низькою відбивною здатністю.
7) Робоче крісло має бути стійким і дозволяти працівникові легко рухатись та займати зручне положення.
8) Сидіння повинно мати систему регулювання висоти і нахилу, за потреби мати підніжку.

Відповідно до НПАОП 0.00-7.15-18 [18] висунуто наступні мінімальні вимоги:

1) Необхідно проводити щоденно прибирання і робочого місця, і самого екранного пристрою.
2) По завершенні роботи із екранним пристроєм, його необхідно вимкнути із електричної мережі.
3) За виникнення аварійної ситуації необхідно негайно вимкнути всі екранні пристрої та всі електричні прилади із електричної мережі.
4) Не допускається:

- Виконання робіт із ремонту, обслуговування або налагодження екранного пристрою під час роботи працівника на робочому місці.
- Вимикати захисні пристрої, власноруч проводити технічні роботи та конструктивні зміни екранного пристрою.

- Працювати із несправним екранним пристроєм чи пристроями, що мають нестабільне зображення або сигналізують про несправність.

5) При виконанні робіт у приміщеннях мають дотримуватись оптимальні умови мікроклімату згідно вимог Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [20].

Відповідно НПАОП 0.00-7.15-18 [18] встановлено наступні мінімальні вимог:

1) Екранні пристрої не мають бути джерелом ризику для працівника.
2) Усі види випромінення, окрім видимої частини електромагнітного спектру, мають бути зведені до незначного рівня з точки зору безпеки та охорони здоров’я працівників.
3) Між символами та рядками повинна бути належна відстань і символи на екранних пристроях повинні бути чіткими й дотримуватись відповідного розміру.
4) Зображення екрану має бути чітким та стабільним, без явних ознак неправильної роботи екранного пристрою.
5) Контрастність та яскравість повинні мати функцію налаштування працівником під час роботи з екранним пристроєм та мати змогу бути швидко адаптованими до навколишнього середовища.
6) При виборі екранного пристрою потрібно враховувати, що він повинен включати в себе функцію налаштування працівником куту нахилу та висоти екрану.
7) За необхідності може використовуватись регульований стіл чи підставка для екрану.
8) Обираючи клавіатуру, слід віддавати перевагу тим клавіатурам, що відокремлені від екранного пристрою та мають функції

налаштування висоти, аби працівник мав змогу зайняти зручну позицію для роботи і уникнути втоми рук.

9) Екран не повинен відбивати світло чи відблискувати під час роботи працівника за екранним пристроєм, щоб не викликати дискомфорту.
10) Поверхня клавіатури не повинна відблискувати чи відбивати світло, щоб уникнути виникнення дискомфорту у працівника. Клавіші та їх розташування мають полегшувати роботу із клавіатурою. Всі клавіші повинні бути позначені контрастними кольорами та бути розбірливими.
11) Прилади, що входять у склад робочої станції не повинні виділяти надлишку тепла для комфортної роботи працівника.
12) Під час роботи працівника з екранними приладами, роботодавець повинен надавати таке програмне забезпечення, що відповідає розв’язуваним працівниками задачам і є простим у використанні і за можливості мати функцію адаптації працівником під себе.

6.2 Шкідливі виробничі фактори на робочому місці

Комп’ютери та інші екранні пристрої з кожним роком все глибше проникають у людське життя і у перед людиною встають питання впливу комп’ютерної техніки на здоров’я. Існує список шкідливих факторів, які діють на людину:

1) Навантаження на зір від екрану
2) Електромагнітні випромінення
3) Сидяче положення на протязі тривалого часу
4) Навантаження на суглоби кистей
5) Навантаження на психічне здоров’я
6) Навантаження на органи слуху за умови гучної роботи ЕОМ

Зазначені вище чинники можуть серйозно вплинути на здоров’я працівника з екранним пристроєм та навіть скоротити термін його життя.

Постійне напруження очей під час роботи за комп’ютером незбіжне, очі постійно дивляться у одну точку та інтервал моргання зменшений, через це слабшають м’язи очей та пересушення кон’юнктиви ока. До того ж на багатьох веб-ресурсах можуть міститись зображення або текст неналежної якості, що через свою погану контрастність і яскравість можуть додатково підвищити навантаження на очі. Для зменшення навантаження на очі рекомендується використовувати правильне освітлення, що освітлює приміщення та обладнання із яким працює працівник, але не екран пристрою, за яким він працює, щоб унеможливити виникнення відблисків, що ускладнюють роботу. Також можна робити гімнастику для очей і обмежувати час проведений перед екраном.

Електромагнітне випромінювання – не менш серйозний чинник загрози здоров’ю. Увімкнений комп’ютер створює поле із широким частотним спектром, що є одним із найнебезпечніших джерел електромагнітного випромінення серед побутових приладів. Від екрану надходить рентгенівське випромінення, а котушки всередині монітору генерують змінне електромагнітне випромінення, що поширюється в різні боки і назад. Також у електронно-променевих трубках висока напруга призводить до появи електростатичного поля за монітором. Однак більшість із цих чинників загрози зведена до мінімуму у сучасній техніці, тому рекомендується використовувати саме її, а також обмежувати час проведений за комп’ютером.

Під час роботи за комп’ютером важливо спостерігати за зручністю пози працівника, адже сидячий образ життя у сукупності із незручним положенням тіла може призводити до багатьох серйозних і небезпечних захворювань. Для зниження ризику цього фактору треба правильно підібрати робочі меблі. Це

допоможе зберігати правильне положення тіла під час роботи за комп’ютером, знизивши м’язове навантаження. Працівник повинен мати змогу переміщати і налагоджувати сидіння під себе для заняття зручної та правильної пози при роботі за комп’ютером.

Робота із мишкою та клавіатурою також можуть призводити до різноманітних захворювань суглобів та м’язів кистей, наприклад тунельного зап’ястного синдрому. Задля зниження ризику таких захворювань потрібно підібрати зручні і ергономічні клавіатуру та миш, які вмістяться на робочому столі і дозволять рукам працівника знаходитись у зручному та безпечному положенні.

Однак не лише тіло може постраждати від роботи за комп’ютером, а й нервова система людини і її мозок. Часто виникають психічні порушення, що є наслідком стресу, виникають тривога, пригніченість і дратівливість у працівників, що працюють за комп’ютерами. В них, частіше ніж у представників інших професій, може спостерігатись безсоння, втрата апетиту, захворювання шкіри рук та обличчя. Для зменшення навантаження на ЦНС рекомендується влаштовувати прогулянки на свіжому повітрі, живе спілкування із людьми та перерви у роботі за комп’ютером.

У таблиці 6.1. наведено допустимі мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт, відповідно до [17].

Таблиця 6.1. – Допустимі мікрокліматичні умови виробничого приміщення, для категорій Іа та Іб легких робіт.

Період року	Категорія робіт	Температура, град.С				Відносна вологість (%)	Швидкість руху, м/сек.
		Верхня мережа		Нижня мережа			
		На постійних робочих місцях	На непостійних робочих місцях	На постійних робочих місцях	На непостійних робочих місцях		
Холодний період року	Легка Іа	25	26	21	18	75	не більше 0,1
	Легка Іб	24	25	21	18	75	Не більше 0,2
Теплий період року	Легка Іа	28	30	22	20	55 – при 28 град.С	0,2-0,1
	Легка Іб	28	30	21	19	60 – при 27 град.С	0,3-0,1

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ВДТ ЕОМ і ПЕОМ [2], мають відповідати вимогам що наведені у таблиці 6.2.

Таблиця 6.2. – Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних смугах частот для програміста

Вид трудової діяльності	Рівні звукового тиску в дБ в октавних смугах із середньогеометричними частотами, Гц									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв.
Програмісти ЕОМ	86	74	61	54	49	45	42	40	38	50

Відповідно до державних будівельних норм ДБН В.2.5-28:2018 «Природне і штучне освітлення», що затверджено наказом Мінрегіону №264 від 3 жовтня 71 2018 року [21] нормативним параметром природного освітлення є коефіцієнт природного освітлення (КПО). Коефіцієнт природного освітлення встановлюється в залежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5-1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для IV розряду зорових робіт мінімальна освітленість складає 300-500 лк.

Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, необхідно визначити світловий потік, що падає на робочу поверхню за формулою (1):

$$F = \frac{ESKZ}{n}, \qquad (1)$$

де F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк Е = 300 Лк;

S – площа освітлюваного приміщення;

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації(К = 1,5)

Z – відношення середньої освітленості до мінімальної (Z =1,1);

n – коефіцієнт використання світлового потоку, (залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін (ρст.) і стелі (ρстелі), значення коефіцієнтів дорівнюють ρст = 30 % і ρстел = 50%).

6.3 Дії працівників у надзвичайних ситуаціях

У випадку аварійної ситуації працівник зобов'язаний:

- У випадку виявлення неполадок в роботі комп'ютера, працівнику потрібно зупинити роботу, виключити комп'ютер та повідомити про неполадку керівника для організації ремонтних робіт.
- При потраплянні людини під дію електричного струму негайно виключити електричне живлення і до прибуття лікаря надати першу медичну допомогу постраждалому.
- За нещасного випадку, отруєння чи раптового захворювання потрібно негайно надати першу медичну допомогу потерпілому, викликати лікаря чи допомогти доставити потерпілого до лікаря, після чого повідомити керівника про те, що сталося.
- При виникненні різі в очах, різкого погіршення зору, виникненні головного болю, больових почуттів у пальцях чи кистях рук, посиленні серцебиття, негайно припинити роботу із ЕОМ, повідомити керівнику про те, що сталось, та звернутись до медичної установи.

- При загорянні технічного обладнання, його потрібно негайно вимкнути із електричної мережі, вжити заходів із ліквідації полум’я за допомогою вогнегасника.

Надання першої медичної допомоги постраждалим від ураження електричним струмом[2]:

1) Якомога швидше відокремити постраждалого від джерела струму.
2) Викликати швидку за необхідності.
3) Покласти та зігріти постраждалого.
4) Закрити опіки, якщо у потерпілого вони є, накрити їх стерильною марлею або чистою гладкою тканиною.
5) За наявності ознак шоку (блювання, слабкість, блідість) – підняти ноги постраждалого, підклавши під ступні валик із речей.
6) Якщо потерпілий погано дихає або зовсім не дихає, негайно розпочати виконання штучного дихання рот у рот.
7) Якщо у потерпілого відсутній пульс і серцебиття, окрім штучного дихання необхідно зробити непрямий масаж серця.

Згідно НАПБ А.01.001-2014, дії із надання першої медичної допомоги у разі пожежі:

1) Якщо горить одяг потерпілого, його слід негайно скинути чи погасити полум’я, щільно накривши потерпілого ковдрою чи яким-небудь шматком тканини. Опалені ділянки одягу обережно розрізати і скидати частинами, аби не травмувати шкіру потерпілого.
2) Якщо у потерпілого наявна закрита рана, уражену ділянку необхідно охолоджувати протягом 10 хвилин.
3) Накласти стерильну пов’язку на поверхню рани.
4) Забезпечити спокій потерпілому.

5) Дати випити велику кількість чаю, води або іншої рідини потерпілому.
6) Викликати бригаду невідкладної допомоги.
7) За можливості знеболити потерпілого, дати йому пігулку анальгіну.

6.4 Висновки за розділом

У ході даного розділу було розглянуто питання зі сфери охорони праці у роботі працівника з екранними пристроями. Розглянуто нормативи і вимоги безпеки до такої праці, можливі шкідливі для здоров’я працівника чинники, характерні для такої роботи, і приведені інструкції щодо поведінки у надзвичайних ситуаціях, таких як пожежа і ураження електричним струмом.

ВИСНОВКИ

У дипломній роботі виконано дослідження та розробку комплексу засобів захищеного обміну повідомленнями на базі симетричного шифрування.

Розроблений комплекс можна використовувати у навчальних цілях для наочного ознайомлення із базовими принципами захищеного обміну інформацією із використанням незахищеного каналу зв’язку.

Розглянуто наявні засоби обміну повідомленнями та порівняно їх між собою з виділенням характерних рис та підходів до захисту інформації.

Описано організацію реалізуємого комплексу. Наведено елементи, з яких він складається та їх призначення. Розроблено структуру бази даних та принципи взаємодії між клієнтом та сервером. Визначено функції сервера та клієнта.

Обрано інструменти для розробки: середовище розробки, мову програмування, систему управління базами даних, бібліотеки, за допомогою яких реалізовано деяку частину функціоналу. На основі алгоритмів розроблено програмне забезпечення програмного комплексу.

Досліджено час виконання криптографічних операцій у залежності від довжини передаваного повідомлення на прикладі блокового симетричного алгоритму та потокового симетричного алгоритму шифрування. За отриманими даними виявлено характерні умови для використання кожного виду алгоритмів.

Сформульовано основні вимоги безпеки при виконанні робіт за електронно обчислювальною машиною. Розглянуто шкідливі фактори, характерні для такого виду виконуваних робіт, та документи, що регулюють ці фактори.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Новини Telegram [Електронний ресурс] – Режим доступу: https://core.telegram.org/blog
2. Захищеність WhatsApp [Електронний ресурс] – Режим доступу: https://www.whatsapp.com/security
3. Захищеність Viber [Електронний ресурс] – Режим доступу: https://www.viber.com/en/security/
4. Новини Skype [Електронний ресурс] – Режим доступу: https://www.skype.com/en/blogs/
5. Про Tox Chat [Електронний ресурс] – Режим доступу: https://tox.chat/faq.html
6. Новини Signal [Електронний ресурс] – Режим доступу: https://www.signal.org/blog/
7. Diffie W., Hellman M. E. New directions in cryptography [Текст] // Information Theory, IEEE Transactions on. — 1976. — нояб. — т. 22, № 6. — с. 644—654. — ISSN 0018-9448. — DOI: 10.1109/TIT.1976.1055638.
8. Черёмушкин А. В. Криптографические протоколы: основные свойства и уязвимости [Текст] // Прикладная дискретная математика. — 2009. — нояб. — вып. 2. — с. 115—150.
9. What is a man-in-the-middle attack? [Електронний ресурс] – Режим доступу: https://us.norton.com/internetsecurity-wifi-what-is-a-man-in-the-middle-attack.html
10. Офіційний сайт Microsoft Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/
11. Документація Python [Електронний ресурс] – Режим доступу: https://www.python.org/doc/

12. Документація SQLite [Електронний ресурс] – Режим доступу: https://sqlite.org/index.html
13. Документація бібліотеки hashlib [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/hashlib.html
14. Фергюсон Н., Шнайер Б. Практическая криптография [Текст]// Компьютерные технологии. – 2016. – 420 с.
15. Документація бібліотеки pyaes [Електронний ресурс] – Режим доступу: https://pypi.org/project/pyaes/
16. Документація бібліотеки RC4 [Електронний ресурс] – Режим доступу: https://github.com/RyanKung/rc4-python3
17. Про охорону праці [Текст]: Закон України згідно з Постановою Верховної Ради України №77-VIII від 28 листопада 2014 року
18. НПАОП 40.1–1.21–98 «Правила безпечної експлуатації електроустановок споживачів» [Текст]: Затверджено: наказ Держнаглядохоронпраці України № 4 від 9 січня 1998 року
19. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час робот з екранними пристроями» [Текст]: Затверджено: наказ Міністерства соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» № 207 від 14 лютого 2018 року
20. ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [Текст]: Затв. Постанова Головного державного санітарного лікаря України від 10 грудня 1998 р. № 7
21. ДСН 3.3.3-042-99 Державні санітарні норми мікроклімату виробничих приміщень [Текст]: Постанова Головного державного санітарного лікаря України від 01.12.1999 №42.

22. ДБН В.2.5-28:2018 «Природне і штучне освітлення» [Текст]: Затверджено наказом Мінрегіону № 264 від 3 жовтня 2018 року
23. ДСН 3.3.6-037-99 Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку» [Текст]: Наказ від 01.12.1999 № 37
24. Ненько С. К., Полівода Л. А. Надання першої медичної допомоги при надзвичайних ситуаціях [Електроний рисурс] / Херсон: «Навчально-методичний центр цивільного захисту та безпеки життєдіяльності Херсонської області», 2014, 28 с. – Режим доступу: https://ks.nmc.dsns.gov.ua/files/documents/first-medical-aid.pdf
25. НПАОП 40.1-1.01-97. Про затвердження Правил безпечної експлуатації електроустановок [Текст]: Нормативно-правовий акт з охорони праці від 6 жовтня 1997 р. № 257

.

ДОДАТОК А. ЛІСТИНГ СЕРВЕРНОЇ ЧАСТИНИ КОМПЛЕКСУ

```
from logging import PlaceHolder
import tkinter as tk
from tkinter import ttk
from tkinter import *
import socket #TCP/IP
from _thread import start_new_thread
import sqlite3
from sys import path
import hashlib
import pyaes
from random import randint
from quopri import encodestring, decodestring
from time import localtime, strftime
global conn
global msg
global active_clients
active_clients = list()
global online_users
online_users = dict()
import pickle

def get_time():
    return strftime("%y-%m-%d %H:%M:%S", localtime())

def msg_db_check():
    cur = db.cursor()
    sql = """create table if not exists Messages (
        Sender varchar(255) not null,
        Text varchar(255) not null,
        Recipient varchar(255) not null,
        Sent_time varchar(255) not null,
        Seen_time varchar(255)

    );"""
    cur.execute(sql)
    db.commit()

def are_keys_created(username, target_user):
    cur = db.cursor()
    sql = f"""select Key from Keyss where Owner = '{username}' and ForUser =
'{target_user}'"""
    cur.execute(sql)
    key = cur.fetchall()
    try:
        key = key[0][0]
    except:
        print('no key found')
    print(key)
    if len(key) > 1:
        print(key)
        return key
    else:
        return 0
```

```
def get_connection():
    pass

def key_exchange(username, target_user):
    p = randint(4294967296 + 1 ,17179869184-1)
    g = randint(4294967296 + 1 ,17179869184-1)
    online_users[username].send(f'GP:{g}:{p}:{target_user}'.encode())
    online_users[target_user].send(f'GP:{g}:{p}:{username}'.encode())

def auth(username, password, connection):
    sql = f"select Password from Users where Username = '{username}'"
    cur = db.cursor()
    cur.execute(sql)
    row = cur.fetchall()
    print(row)
    if len(row) != 0:
        provided_password = row[0][0]
        password = password.encode()
        password = hashlib.sha256(password)
        password = password.hexdigest()
        if password == provided_password:
            print(f"{username}: AUTH successful")
            connection.send("AUTH Code: 1".encode())
            online_users[username] = connection
        else:
            error = "Incorrect password"
            connection.send(error.encode())
            print("Incorrect password")
    else:
        error = "No such user"
        connection.send(error.encode())
        print("No such user")

def add_new_client():
    sql = """create table if not exists Users (
        Username varchar(255) not null,
        Password varchar(255) not null

    );"""
    cur = db.cursor()
    cur.execute(sql)
    username = username_input.get()
    password = password_input.get()
    password = password.encode()
    password = hashlib.sha256(password)
    password = password.hexdigest()
    cur = db.cursor()
    sql = f"select * from Users where Username='{username}'"
    cur.execute(sql)
    rows = cur.fetchall()
    if len(rows) >= 1:
        print("Such user already exists")
    else:
        sql = f"""insert into Users(Username, Password)
```

```
            VALUES('{username}','{password}')"""
        print(sql)
        history.insert(END, f'User {username} was created succesfully')
        cur = db.cursor()
        cur.execute(sql)
        db.commit()

def list_users():
    sql = """select * from Users;"""
    cur = db.cursor()
    cur.execute(sql)
    rows = cur.fetchall()
    for row in rows:
        print(row)

def list_online():
    for user in online_users.keys():
        print(user)
        return online_users.keys()


def wait_for_clients():
    while True:
        print('Entered wait_for_clients()')
        ServerSocket.listen(5)
        Client, address = ServerSocket.accept()
        print('Connected to: ' + address[0] + ':' + str(address[1]))
        start_new_thread(threaded_client, (Client, ))

def start():
    sql = "delete from Messages"
    cur = db.cursor()
    cur.execute(sql)
    global ServerSocket
    ServerSocket = socket.socket()
    host = server_ip_input.get()
    port = int(server_port_input.get())
    try:
        ServerSocket.bind((host, port))
    except socket.error as e:
        print(str(e))

    print('Waiting for a Connection..')
    msg_db_check()
    start_new_thread(wait_for_clients, ())
    server_ip_input.configure(state=tk.DISABLED)
    server_port_input.configure(state=tk.DISABLED)

def threaded_client(connection):
    active_clients.append(connection)
    while True:
        data = connection.recv(2048).decode()
        print('received data')
        print(data)
        if 'AUTH' in data:
```

```
            print(f'{connection} is authenticating')
            data = data[5:]
            data = data.split(':')
            username = data[0]
            password = data[1]
            history.insert(END, f'{username} is authenticating')
            auth(username, password, connection)
        elif 'ONLINE' in data:
            users = 'ONLINE:'
            for user in list_online():
                users += f'{user},'
                connection.send(users.encode())
        elif 'CONNECT' in data:
            target_user = data[8:]
            key = are_keys_created(username, target_user)
            if key == 0:
                print(f'User {username} has no key for {target_user}')
                key_exchange(username, target_user)
            else:
                print(data)
                online = list_online()
                print(online)
                users = list()
                print(users)
                for user in online:
                    print(user)
                    print(target_user)
                    i = 0
                    if user == target_user:
                        connection.send(f'KEY:{key}'.encode())
                        online_users[target_user].send(f'DIALOG
INIT:{username}'.encode())
                        print('Send dialog init message')
                        print(online_users)
                        i = 1
                if i == 0:
                    connection.send('User is offline or does not exist'.encode())

        elif 'A:' in data:
                A = data
                A = A.split(':')
                target_user = A[1]
                A = A[2]
                print(f'{username} sent A to {target_user}')
                online_users[target_user].send(f'A:{A}'.encode())
                print('Key exchange successful')
        elif 'MSG' in data:
            data = data.split(':')
            print(data)
            another_user = data[1]
            online_users[another_user].send(f'MSG:{username}'.encode())
            text = connection.recv(2048)
            online_users[another_user].send(text)
            print(text)
            text = int.from_bytes(text, 'big')
```

```
            print(text)
            curr_time = get_time()
            sql = f"""insert into Messages(Sender, Text, Recipient, Sent_time)
                values ('{username}', '{text}', '{another_user}', '{curr_time}');"""
            cur = db.cursor()
            cur.execute(sql)
            print('Message object created in DB')
        elif 'REFRESH' in data:
            data = data.split(':')
            username = data[1]
            another_user = data[2]
            curr_time = get_time()
            sql = f"""update Messages set Seen_time = '{curr_time}'
            where Sender = '{another_user}' and Recipient = '{username}';"""
            cur = db.cursor()
            cur.execute(sql)
            db.commit()

            sql = f"""select * from Messages where ((Sender = '{username}'
             and Recipient = '{another_user}') or (Sender = '{another_user}'
             and Recipient = '{username}'));"""
            cur = db.cursor()
            cur.execute(sql)
            rows = cur.fetchall()
            print(rows)
            rows = pickle.dumps(rows)
            connection.send(rows)


        elif 'K:' in data:
            print('writing key to db')
            data = data.split(':')
            username = data[1]
            print(username)
            target_user = data[2]
            print(target_user)
            ek = connection.recv(2048)
            ek = int.from_bytes(ek, 'big')
            ek = str(ek)
            print(type(ek))
            print(ek)
            sql    =    f"insert    into    Keyss(Owner,    ForUser,    Key)    values
('{username}','{target_user}', '{ek}');"
            cur = db.cursor()
            cur.execute(sql)
            db.commit()
            sql = "select * from Keyss;"
            cur = db.cursor()
            cur.execute(sql)
            rows = cur.fetchall()
            rows = pickle.dumps(rows)
            connection.send(rows)
        elif not data:
            break
    connection.close()
```

```
    active_clients.remove(connection)
    online_users.pop(username)

def send(msg, connection):
    connection.send(msg)
    msg = msg.decode()
    print(msg)
    msg = 'Server: ' + str(msg)
    history.insert(END, msg)


root = Tk()

root.geometry('652x571')
root.configure(background='#F0F8FF')
root.title('Server')

Label(root, text='Server IP', bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=30,
y=30)
server_ip_input = Entry(root)
server_ip_input.place(x=120, y=30)
server_ip_input.insert(0, '192.168.0.120')

Label(root,     text='Server     Port',     bg='#F0F8FF',     font=('arial',     12,
'normal')).place(x=30, y=60)
server_port_input = Entry(root)
server_port_input.place(x=120, y=60)
server_port_input.insert(0, '5000')

connect_button = Button(root, text='Start', command=start)
connect_button.place(x=90, y=90)

stop_button = Button(root, text='Stop' )
stop_button.place(x=150, y=90)

history = Listbox(root)
history.place(x=30, y=150, width=200)

username_input = Entry(root)
username_input.place(x=30, y=120)

password_input = Entry(root)
password_input.place(x=130, y=120)

new_user_button = Button(root, text="Add new client", command=add_new_client)
new_user_button.place(x=230, y=120)

global db
db = sqlite3.connect('database.db', check_same_thread=False)

root.mainloop()
db.close()
```

ДОДАТОК Б. ЛІСТИНГ КЛІЄНТСЬКОЇ ЧАСТИНИ КОМПЛЕКСУ

```
from logging import PlaceHolder
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from tkinter import *
from threading import Thread
import socket
import hashlib
import pyaes
from random import randint
import configparser
from time import sleep
import pickle

global k

class ReplyHandler(Thread):
    def __init__(self):
        Thread.__init__(self)
    def run(self):
        while True:
            reply = sock.recv(1024)
            try:
                check = reply.decode()
            except:
                print("Reply undecodable")
                check = "Something else"
            print(reply)
            if check == "AUTH Code: 1":
                reply = reply.decode()
                print("authentication successful")
                username_input.configure(state=tk.DISABLED)
                password_input.configure(state=tk.DISABLED)
            elif 'ONLINE' in check:
                reply = reply.decode()
                reply = reply[7:]
                reply = reply.split(',')
                users.delete(0,END)
                for user in reply:
                    if len(user) > 2:
                        users.insert(END, user)
            elif "DIALOG INIT" in check:
                reply = reply.decode()
                print("Dialog initiated")
                global another_user
                another_user = reply[12:]
                print(f'talking to {another_user}')
                another_user_label.configure(text=f'{another_user}')
            elif 'GP' in check:
                reply = reply.decode()
                print(reply)
                reply = reply.split(':')
                print(reply)
```

```
                g = int(reply[1])
                p = int(reply[2])
                recipient = reply[3]
                a = randint(4294967296 + 1 ,17179869184-1)
                A = pow(g,a,p)
                sock.send(f'A:{recipient}:{A}'.encode())

            elif "A:" in check:
                reply = reply.decode()
                print(reply)
                reply = reply.split(":")
                print(reply)
                B = int(reply[1])
                k = pow(B,a,p)
                print(k)
                epassword = ''
                for c in password:
                    epassword += str(ord(c))
                epassword = int(epassword)
                epassword = epassword.to_bytes(32, 'big')
                aes_for_key = pyaes.AESModeOfOperationCTR(epassword)
                k = str(k)
                ek = aes_for_key.encrypt(k)
                k = int(k)
                cp = configparser.RawConfigParser()
                cp_path = "user_creds.txt"
                cp.read(cp_path)
                cp.set('creds','key', k)
                k = k.to_bytes(32, 'big')
                aes = pyaes.AESModeOfOperationCTR(k)
                print(k)
                print(ek)
                another_user = recipient
                sock.send(f'K:{username}:{recipient}'.encode())
                sock.send(ek)
                print('Key exchange successful')
                another_user_label.configure(text=f'You are chatting with
{recipient}')
            elif 'KEY' in check:
                reply = reply.decode()
                reply = reply.split(':')
                k = reply[1]
                k = int(k)
                cp = configparser.RawConfigParser()
                cp_path = "user_creds.txt"
                cp.read(cp_path)
                cp.set('creds','key', k)
                k = k.to_bytes(32, 'big')
                aes = pyaes.AESModeOfOperationCTR(k)
            elif 'MSG' in check:
                print(reply)
                reply = reply.decode()
                print(reply)
                message = reply.split(':')
                another_user = message[1]
```

```
                print(f"REFRESH:{username}:{another_user}".encode())
            else:
                i = 0
                try:
                    histoire = pickle.loads(reply)
                    i = 1
                except:
                    print("It's not history")
                if i == 1:
                    history.delete(0,END)
                    for row in histoire:
                        message = f'{row[0]}:'
                        cp = configparser.RawConfigParser()
                        cp_path = "user_creds.txt"
                        cp.read(cp_path)
                        k = cp.get('creds', 'key')
                        k = int(k)
                        k = k.to_bytes(32, 'big')
                        aes = pyaes.AESModeOfOperationCTR(k)
                        text = int(row[1])
                        text = text.to_bytes(text.bit_length(),'big')
                        text = aes.decrypt(text)
                        message += f'{text}:'
                        message += f' [{row[3]}]'
                        try:
                            if row[4] > row[3]:
                                message += " (seen)"
                        except:
                            print("Seen time is probably null")
                        history.insert(END, message)
                    print("Ended appending history")
                else:
                    text = reply
                    print(text)
                    print(f"Received text = {text}")
                    cp = configparser.RawConfigParser()
                    cp_path = "user_creds.txt"
                    cp.read(cp_path)
                    k = cp.get('creds', 'key')
                    k = int(k)
                    k = k.to_bytes(32, 'big')
                    aes = pyaes.AESModeOfOperationCTR(k)
                    print(text)
                    text = aes.decrypt(text)
                    print(f'Decrypted text = {text}')
                    text = str(text)
                    text = text[2:-1]
                    print(another_user)
                    print(text)
                    history.insert(END, another_user + ': ' + text)

def refresh_history(username, another_user):
    print(f"REFRESH:{username}:{another_user}".encode())
    iterations = sock.recv(2048).decode()
    iterations = int(iterations)
```

```
    i = 1
    history.delete(0,END)
    if iterations != 0:
        while i != iterations:
            reply = sock.recv(2048).decode()
            print(reply)
            reply = reply.split('!')
            print(reply)
            sender = reply[0]
            text = sock.recv(2048)
            cp = configparser.RawConfigParser()
            cp_path = "user_creds.txt"
            cp.read(cp_path)
            k = cp.get('creds', 'key')
            k = int(k)
            k = int.to_bytes(k, 32, 'big')
            aes = pyaes.AESModeOfOperationCTR(k)
            text = aes.decrypt(text)
            text = str(text)
            sent_time = reply[3]
            message = f'{sender}: {text} [{sent_time}]'
            if len(reply[4]) > 1:
                message += ' (seen)'
            history.insert(END, message)
            i += 1


def enter_dialog():
    print(username)
    global another_user
    another_user = enter_dialog_with_input.get()
    print(another_user)
    if another_user == username:
        print("You can't talk to yourself")
    else:
        sock.send(f'CONNECT:{another_user}'.encode())
        another_user_label.configure(text=f'You are chatting with {another_user}')
        enter_dialog_with_input.delete(0, END)

def send_message():
    pass

def get_online():
    sock.send('ONLINE'.encode())


def connect(host='192.168.0.120', port = 5000):
   global sock
   sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   sock.connect(('192.168.0.120', 5000))
   thread = ReplyHandler()
   thread.daemon = True
   thread.start()
   global username
   global password
```

```
   username = username_input.get()
   password = password_input.get()
   sock.send(f'AUTH:{username}:{password}'.encode())

def add_contact():
    pass

def send():
    cp = configparser.RawConfigParser()
    cp_path = "user_creds.txt"
    cp.read(cp_path)
    k = cp.get('creds', 'key')
    k = int(k)
    k = k.to_bytes(32, 'big')
    message = 'MSG:'+another_user
    print(message)
    sock.send(message.encode())
    text = message_entry.get()
    aes = pyaes.AESModeOfOperationCTR(k)
    text = aes.encrypt(text)
    print(text)
    sock.send(text)
    history.insert(END, username + ': ' + message_entry.get())
    message_entry.delete(0, END)


root = Tk()

root.geometry('900x571')
root.configure(background='#F0F8FF')
root.title('Client')

Label(root, text='Username', bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=30,
y=90)
username_input = Entry(root)
username_input.place(x=120, y=90)
username_input.insert(0, 'testuser')

Label(root, text='Password', bg='#F0F8FF', font=('arial', 12, 'normal')).place(x=30,
y=120)
password_input = Entry(root, show='*')
password_input.place(x=120, y=120)
password_input.insert(0,'123')

connect_button = Button(root, text='Connect', command=connect)
connect_button.place(x=30, y=210)

disconnect_button = Button(root, text='Disconnect')
disconnect_button.place(x=90, y=210)

enter_dialog_with_input = Entry(root)
enter_dialog_with_input.place(x=160, y=240)

enter_dialog_with_button = Button(root, text="Enter dialog", command=enter_dialog)
enter_dialog_with_button.place(x=160, y=270)
```

```
users = Listbox(root)
users.place(x=30, y=270)

get_online_button = Button(root, text="Get Online Users", command=get_online)
get_online_button.place(x=30, y=240)

another_user_label = Label(root, text='', bg='#F0F8FF', font=('arial', 12, 'normal'))
another_user_label.place(x=290, y=210)

history = Listbox(root, width=100)
history.place(x=290, y=240)

message_entry = Entry(root)
message_entry.place(x=290, y=410)

send_button = Button(root, text="Send", command=send)
send_button.place(x=410, y=410)

try:
    cp = configparser.RawConfigParser()
    cp_path = "user_creds.txt"
    cp.read(cp_path)
    usr = cp.get('creds', 'username')
    pasw = cp.get('creds', 'password')
    username_input.delete(0,END)
    username_input.insert(0, usr)
    password_input.delete(0,END)
    password_input.insert(0, pasw)
except:
    pass

root.mainloop()
```