Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана»

Виконавець: студент IV курсу, групи КБ-41

_____ **Деніс РОСІЙСЬКОВ**
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник	Сергій ДАКОВ	

Нормоконтроль	Юрій ЩЕБЛАНІН	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-41** **Російськову Денісу Сергійовичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Захист інформації, симетричні та асиметричні алгоритми шифрування

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Захист інформації за допомогою шифрування
Основні компоненти програмного забезпечення
Озробка програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблене програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29 листопада 2021 року

Завдання видав _____ Сергій ДАКОВ
(підпис) (ім'я, прізвище)

Завдання прийняв _____ Деніс РОСІЙСЬКОВ
до виконання (підпис) (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.10.2021 – 03.11.2021	*виконано*
2	Аналіз літератури	04.11.2021 – 01.12.2021	*виконано*
3	Розгляд криптографічних алгоритмів шифрування	02.01.2022 – 17.01.2022	*виконано*
4	Ознайомлення зі структурою клієнт-серверної архітектури	18.01.2022 – 06.02.2022	*виконано*
5	Дослідження роботи криптографічного протоколу Діффі-Геллмана	07.02.2022 – 20.02.2022	*виконано*
6	Побудова архітектури програмного забезпечення	21.02.2022 – 27.03.2022	*виконано*
7	Вибір інструментів для створення програмного забезпечення	28.03.2022 – 05.04.2022	*виконано*
8	Розробка та тестування програмного забезпечення	06.04.2022 – 28.05.2022	*виконано*
9	Оформлення пояснювальної записки	29.05.2022 – 01.06.2022	*виконано*
10	Підготовка до захисту	02.06.2022 – 10.06.2022	*виконано*

Завдання видав _____ Сергій ДАКОВ
(підпис) (ім'я прізвище)

Завдання прийняв _____ Деніс РОСІЙСЬКОВ
до виконання (підпис) (ім'я прізвище)

Термін подання дипломної роботи до ЕК 06 червня 2022 року

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 58 сторінок. Крім того, робота містить 1 додаток із загальною кількістю сторінок 9. У пояснювальній записці дипломної роботи міститься 37 рисунків.

Метою роботи є розробка програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Для досягнення зазначеної мети поставлено наступні завдання:

- дослідити криптографічні алгоритми шифрування, їх види, переваги та недоліки;
- розглянути проблему обміну ключами при використанні симетричних алгоритмів шифрування та можливі шляхи її вирішення;
- побудувати архітектуру програмного забезпечення, визначити основні необхідні компоненти для його створення.
- описати роботу використаних бібліотек, створених класів, методів та функцій з використанням схем.
- розробити програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана;
- протестувати роботу створеного програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

В роботі проведено аналіз захисту інформації за допомогою шифрування.

Запропоновано використання асиметричної криптографії для обміну секретними ключами при використанні незахищених каналів зв’язку на базі криптографічного протоколу Діффі-Геллмана.

Побудовано модель програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Розроблено програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Об’єктом дослідження є процес обміну ключами за допомогою криптографічних протоколів.

Предметом дослідження є механізми забезпечення безпечного обміну ключами при використанні незахищених каналів зв’язку.

Методи дослідження: системний підхід, структурний аналіз.

Практичною цінністю отриманих результатів є розроблене програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Результати здійснених у дипломній роботі досліджень можуть бути використані в програмних продуктах для забезпечення безпечного обміну ключами.

Напрямки подальших досліджень: вивчення більш надійних і захищених симетричних алгоритмів шифрування.

Ключові слова: захист інформації, клієнт-серверна архітектура, шифрування, симетричні алгоритми шифрування, асиметричні алгоритми шифрування, криптографічний протокол Діффі-Геллмана, обмін ключами.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VPN	–	Virtual Private Network
IDE	–	Integrated Development Environment
ПЗ	–	Програмне Забезпечення
OSI	–	Open Systems Interconnection
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
ОС	–	Операційна Система

ЗМІСТ

РЕФЕРАТ 4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6
ЗМІСТ 7
ВСТУП 9
РОЗДІЛ 1 ЗАХИСТ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ШИФРУВАННЯ 10
1.1 Криптографічні алгоритми шифрування 10
1.2 Технологія VPN 12
1.3 Алгоритми обміну ключами з використанням незахищених каналів зв’язку 14
1.4. Клієнт-серверна архітектура 15
1.5. Постановка задачі 16
Висновки за розділом 1 16
РОЗДІЛ 2 ОСНОВНІ КОМПОНЕНТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 18
2.1 Архітектура програмного забезпечення 18
2.2 Модель криптографічного протоколу Діффі-Геллмана 20
2.3 Модель шифру Цезаря 21
2.4 Редактор коду Visual Studio Code 22
2.5 Мова програмування Python 24
2.6 Технології віртуалізації для тестування програмного забезпечення 26
Висновки за розділом 2 28
РОДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБМІНУ СЕКРЕТНИМИ КЛЮЧАМИ НА БАЗІ ПРОТОКОЛУ ДІФФІ-ГЕЛЛМАНА 29
3.1 Загальний опис та структура програмного забезпечення 29
3.2 Опис використаних модулів 30
3.3 Функції та методи для роботи з базовими типами 31
3.4 Опис класів та функцій 32
3.5 Опис роботи сервера та клієнта 44
3.6 Тестування програмного забезпечення 48
Висновки за розділом 3 55

ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТКИ 59

ВСТУП

Актуальність даної роботи пов’язана з активним розвитком мережі Інтернет та популяризацією використання корпоративних мереж. В більшості випадків, інформація передається відкритими каналами зв’язку, які є незахищеними і можуть прослуховуватися зловмисниками з метою порушення конфіденційності інформації, яка циркулює в таких каналах.

Одним з основних методів захисту інформації є шифрування, яке дозволяє захистити інформацію від можливого несанкціонованого ознайомлення. Найчастіше для шифрування інформації використовують симетричні алгоритми шифрування, так як вони працюють швидше за асиметричні, є нескладними у реалізації і більше підходять для шифрування великих обсягів інформації. Але, при використанні симетричних алгоритмів шифрування суттєвою задачею є забезпечення безпечного обміну ключами між користувачами. Дана проблема пов’язана з тим, що для шифрування та розшифрування використовується один і той самий ключ і, в разі його компрометації, зловмисник може отримати доступ до захищеної інформації. Так як ключі не можна передавати відкритими каналами зв’язку, а створення власного захищеного каналу зв’язку вимагає великих витрат, то виникла необхідність створення механізмів які б могли забезпечити безпечний обмін ключами між користувачами при використанні незахищених каналів зв’язку. Так з’явилися спеціальні криптографічні протоколи, які вирішують дану задачу за допомогою асиметричної криптографії. Одним з таких протоколів є протокол Діффі-Геллмана, який розглядається в даній роботі.

Метою роботи є розробка програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Об’єктом дослідження є процес обміну ключами за допомогою криптографічних протоколів.

Предметом дослідження є механізми забезпечення безпечного обміну ключами при використанні незахищених каналів зв’язку.

РОЗДІЛ 1
ЗАХИСТ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ШИФРУВАННЯ

1.1 Криптографічні алгоритми шифрування

Найчастіше для захисту інформації під час її передачі використовують криптографічні алгоритми, які забезпечують збереження однієї з основних властивостей інформації – конфіденційності.

Існує велика кількість різних криптографічних алгоритмів. Основою будь-якого криптографічного алгоритму є ключі, які використовують для шифрування (розшифрування) інформації. В залежності від того, чи співпадає ключ для шифрування з ключем для розшифрування, криптографічні алгоритми поділяють на дві великі групи:

- симетричні алгоритми шифрування;
- асиметричні алгоритми шифрування.

Симетричні алгоритми шифрування

Симетричні алгоритми шифрування – це криптографічні алгоритми в яких для шифрування і розшифрування даних використовуються однакові ключі.

Переваги використання симетричних алгоритмів шифрування [1]:

- Швидкість роботи. Симетричні алгоритми шифрування швидше працюють за асиметричні шифрування.
- Простота реалізації з рахунок використання простих операцій.
- Довжина ключа. Симетричні алгоритми шифрування не потребують відносно довгої довжини ключа.

До недоліків використання симетричних алгоритмів шифрування можна віднести [1]:

- Складність управління ключами у великих мережах.
- Складність обміну ключами, яка полягає у забезпеченні безпечного процесу обміну ключами між абонентами мережі з метою збереження їх секретності.

Схему роботи симетричних алгоритмів шифрування наведено на рисунку 1.1.

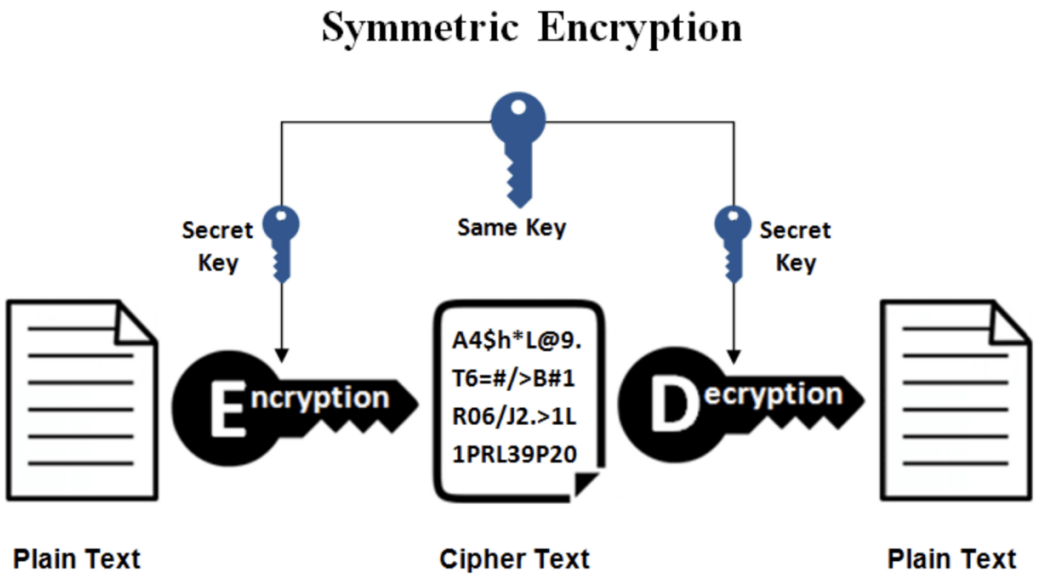


Рисунок 1.1 Симетричне шифрування

Асиметричні алгоритми шифрування

Симетричні алгоритми шифрування – це криптографічні алгоритми в яких для шифрування і розшифрування даних використовуються різні ключі, один з яких є публічним, який відомий всім, а інший – приватним, який відомий лише власнику цього ключа.

До переваг використання асиметричних алгоритмів шифрування можна віднести:

- Безпека. Асиметричні алгоритми шифрування є надійнішими з точки зору безпеки, ніж симетричні.
- Пару публічний / секретний ключ можна значний час не змінювати.
- Не потрібно передавати особистий (приватний) ключ захищеними каналами зв’язку.

Недоліками використання асиметричних алгоритмів шифрування є:

- Реалізація асиметричних алгоритмів шифрування потребує більших обчислювальних ресурсів, ніж симетричних.

- Як правило, довжина ключів в асиметричних алгоритмах шифрування довша, ніж в симетричних.
- Складність вносити зміни до алгоритму.

Часто, для компенсації недоліків симетричних та асиметричних алгоритмів використовують гібридні системи, в яких асиметрична криптографія використовуються для обміну ключами між абонентами, а симетрична – для безпечного обміну даними.

Схему роботи асиметричних алгоритмів шифрування наведено на рисунку 1.2.

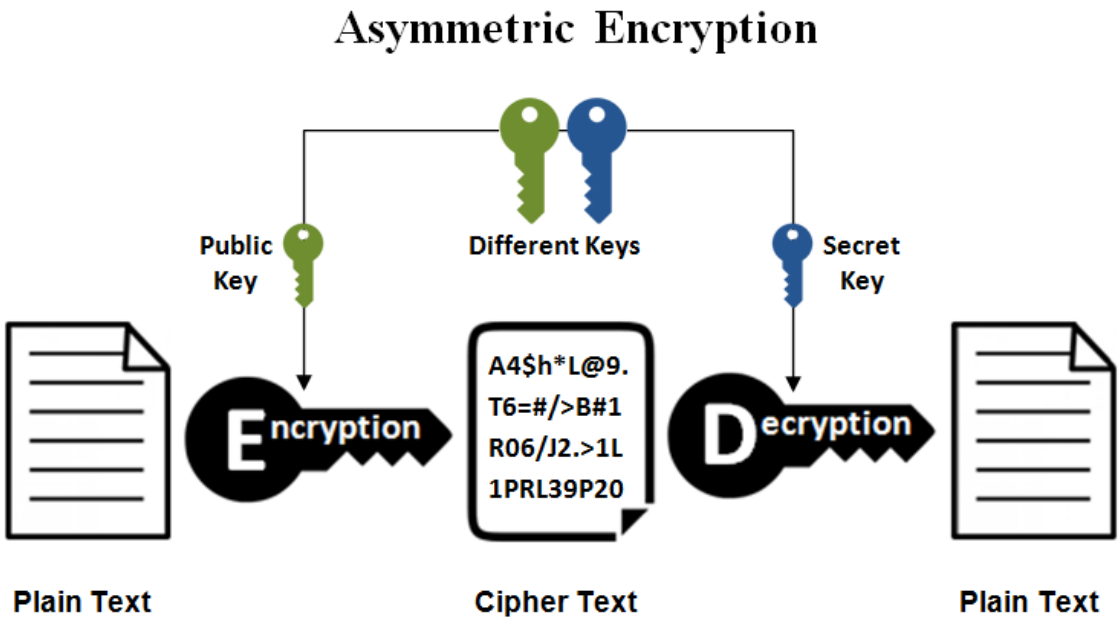


Рисунок 1.2 Асиметричне шифрування

1.2 Технологія VPN

VPN (віртуальна приватна мережа) – це безпечне, зашифроване з'єднання через Інтернет між користувачем і мережею або двома мережами [2]. Використання VPN дозоляє зберегти конфіденційність – одну з трьох основних властивостей інформації. Також VPN використовують для збереження анонімності в мережі.

Віртуальна приватна мережа забезпечує безпечну передачу трафіку за допомогою встановлення захищеного з'єднання через Інтернет, яке ще називають тунелем. Тунель забезпечує шифрування даних, які через нього проходять.

Загалом, виділяють два основних види мереж VPN:

- Шлюз безпечного віддаленого доступу, який дозволяє користувачам підключатися до мережі Інтернет (корпоративної мережі тощо) за допомогою приватного зашифрованого тунелю (рис 1.3).

Рисунок 1.3 Шлюз безпечного віддаленого доступу

- VPN між маршрутизаторами (VPN типу «мережа-мережа»), який використовується для створення закритої внутрішньої мережі. Даний вид часто використовується в корпоративному середовищі (рис. 1.4).

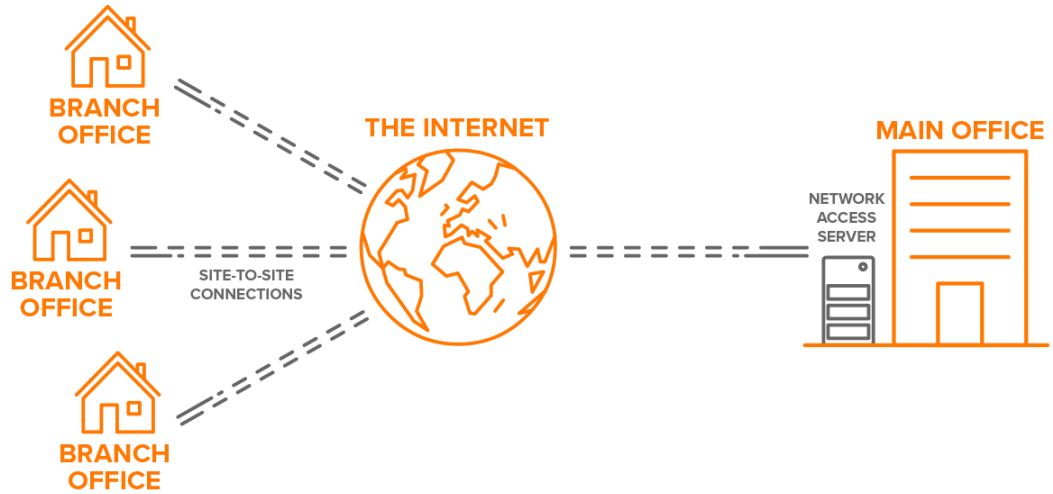


Рисунок 1.4 VPN типу «мережа-мережа»

Переваги використання VPN:

- Збереження конфіденційності та анонімності.

- Обхід цінової дискримінації.
- Отримання доступу до заблокованих веб-сайтів.

Недоліки використання VPN:

- Витрати. Безкоштовні VPN не забезпечують такої надійності та безпеки, як платні.
- Деякі провайдери VPN реєструють користувацькі дані, що є загрозою конфіденційності.
- Сповільнення роботи. Використання VPN зменшує швидкість роботи в мережі.
- Відсутність повної конфіденційності.

1.3 Алгоритми обміну ключами з використанням незахищених каналів зв'язку

Перед тим як здійснювати захист інформації за допомогою шифрування, сторонам, які будуть обмінюватися цією інформацією, необхідно здійснити обмін ключами. Організація безпечного обміну ключами зі збереженням їх секретності є однією з основних задач при використанні симетричних алгоритмів шифрування. В реальних умовах користувачі, яким необхідно обмінятися ключами, можуть знаходитися на великих відстанях один від одного і навіть не знати один одного, що стає проблемою для реалізації безпечного обміну ключами. Ключі не можна відправляти звичайними способами, так як зловмисники можуть їх перехопити і, тим самим, будуть мати можливість отримати несанкціонований доступ до даних. Для вирішення такої проблеми було створено спеціальні криптографічні протоколи, які дозволяють сторонам безпечно обмінятися ключами з використанням незахищених каналів зв'язку, таких як Інтернет.

Найбільш популярними криптографічними протоколами для обміну ключами є протокол Діффі-Геллмана та протокол RSA. Поява цих двох протоколів значно прискорила розвиток Інтернету. Пов'язано це було з тим, що саме ці два протоколи

могли забезпечити безпечний обмін ключами при використанні незахищених каналів зв’язку, які можуть прослуховуватися зловмисниками [3].

1.4. Клієнт-серверна архітектура

Архітектура клієнт–сервер (client-server architecture) – це концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена на серверах, обслуговуючих своїх клієнтів. Дана архітектура визначає три типи компонентів [4, 5]:

- Сервер – об’єкт, який надає послуги та ресурси (сервіс) клієнтам.
- Клієнт – об’єкт, який використовує послуги та ресурси (сервіс), які надає сервер.
- Мережа, яка забезпечує взаємодію між клієнтом та сервером. Взаємодія між клієнтом та сервером відбувається за певним набором правил, які називаються протоколом обміну (протоколом взаємодії).

Сервер надає сервіс клієнтам. Він отримує запити від клієнтів на виконання певних завдань, виконує їх та надсилає відповідь з результатами виконання. Клієнт, отримавши відповідь від сервера, подає її у зручному форматі кінцевому користувачу завдяки інтерфейсам користувача. Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше [6].

В залежності від розподілу функцій між клієнтом та сервером розрізняють дві моделі:

- Модель товстого клієнта, коли більшість функціоналу покладена на клієнта.
- Модель тонкого клієнта, коли виконання більшості операцій та функцій покладено на сервер. В даному випадку роль клієнта полягає лише в представленні даних користувачу, а основні обчислення відбуваються на сервері.

Спрощене представлення клієнт серверної архітектури зображено на рисунку 1.5.

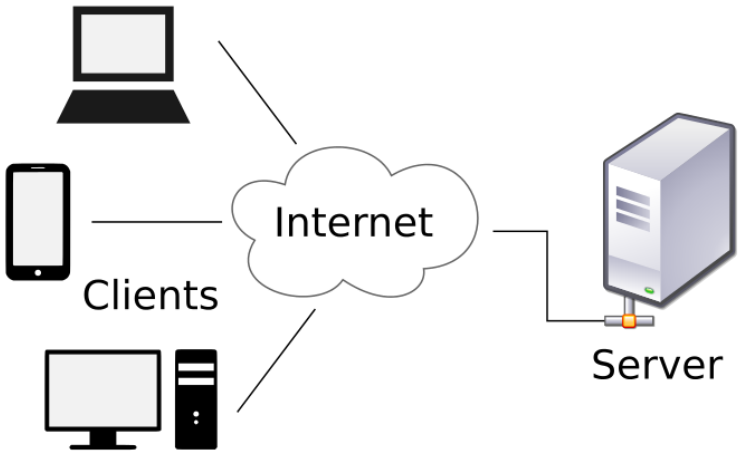


Рисунок 1.5 Клієнт-серверна архітектура

1.5. Постановка задачі

Відповідно до теми дипломної роботи, поставленою задачею було розробити програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана. Для виконання поставленої задачі було вирішено створити програмне забезпечення з використанням клієнт-серверної архітектури, яке реалізовує процес ідентифікації, автентифікаціїї та авторизації клієнта на сервері. В якості інформації, яку потрібно захищати під час її передачі від клієнта до сервера за допомогою шифрування було обрано ідентифікаційні (логіни) та автентифікаційні (паролі) дані клієнтів. Ключі для шифрування (розшифрування) отримуються за допомогою криптографічного протоколу Діффі-Геллмана безпосередньо перед відправленням ідентифікаційних та автентифікаційних даних від клієнта до сервера.

Висновки за розділом 1

Проаналізовано теоретичні відомості про криптографічні алгоритми шифрування, а саме про симетричні та асиметричні алгоритми шифрування, їх переваги та недоліки, можливості поєднання цих алгоритмів з метою компенсації їх

недоліків. Було описано основну складність використання симетричних алгоритмів шифрування та можливості її вирішення.

Описано технологію VPN, яка забезпечує безпечний канал зв’язку між користувачами або маршрутизаторами, який називається тунелем.

У сучасному, в основному, в якості абонентів, яким необхідно обмінятися ключами, виступають клієнт та сервер. У зв’язку з цим було описано основні особливості клієнт-серверної архітектури.

В даному розділі також було описано основну задачу, яка полягає в розробці програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

РОЗДІЛ 2
ОСНОВНІ КОМПОНЕНТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура програмного забезпечення

Розробка архітектури програмного забезпечення включає в себе визначення структури створюваного програмного забезпечення, його компонентів та те, як вони між собою пов’язані, які задачі вирішують і як взаємодіють між собою. Архітектура програмного забезпечення є своєрідним шаблоном, за яким створюється програмний продукт. Вона надає уявлення про те, яким чином буде працювати створюване програмне забезпечення, з яких основних компонентів воно буде складатися і які задачі вирішуватиме. При розробці архітектури програмного забезпечення враховувались наступні критерії:

- Модульність – програмне забезпечення має складатися з окремих незалежних один від одного модулів.
- Мінімізація повторів – зменшення кількості повторюваних операцій за рахунок створення класів, методів та функцій з метою подальшого їх виклику.
- Ефективність – програмне забезпечення має виконувати поставлену задачу: реалізація процесу безпечного обміну секретними ключами за допомогою протоколу Діффі-Геллмана при використанні незахищених каналів зв’язку.
- Гнучкість та масштабованість – програмне забезпечення може з часом змінюватися, модифікуватися тощо.

В основі розробленої архітектури лежить клієнт-серверна архітектура. Розроблене програмне забезпечення реалізує механізм обміну секретними ключами між клієнтом та сервером за допомогою протоколу Діффі-Геллмана, які потім використовуються з метою шифрування інформації, якою обмінюються клієнт та сервер. Відповідно до поставлених задач програмне забезпечення буде складатися з трьох частин:

- серверної, яка буде описувати роботу сервера.

- клієнтської, яка буде описувати роботу клієнта.
- функціональної, яка буде зосереджувати в собі основні функції, які будуть використовуватися в серверній та клієнтській частинах.

Архітектуру створеного програмного забезпечення наведено на рисунку 2.1.

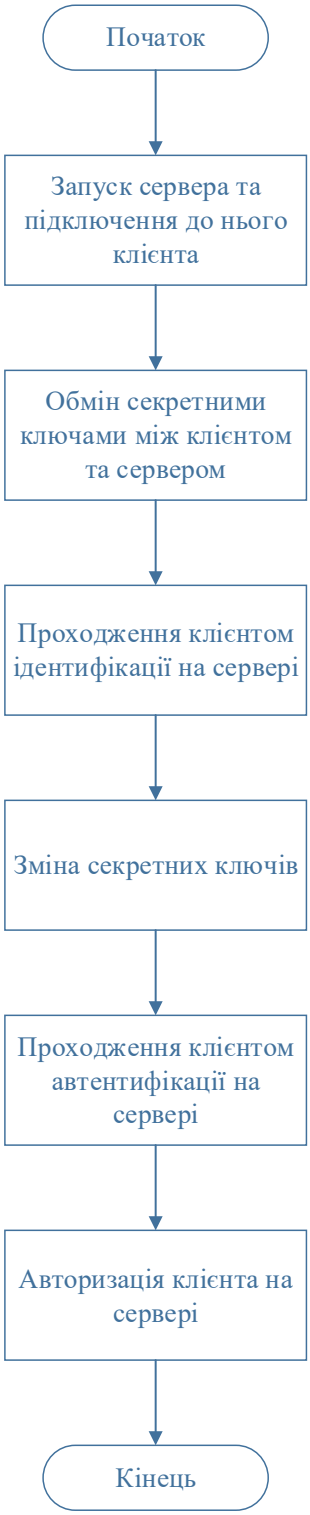


Рисунок 2.1 Архітектура програмного забезпечення

2.2 Модель криптографічного протоколу Діффі-Геллмана

Протокол Діффі-Геллмана – це криптографічний протокол, який дозволяє двом сторонам безпечно обмінятися секретними криптографічними ключами з використанням незахищеного каналу зв'язку, які потім можуть бути використані в симетричних алгоритмах шифрування. Обчислення спільного секретного ключа відбувається з використанням попередньо обраних двох публічних ключів, які відомі всім бажаючим, та двох приватних ключів, які відомі лише їх власникам. Процес обміну секретними ключами можна описати наступним чином:

- Нехай є два користувачі – Аліса та Боб, яким необхідно отримати спільний секретний ключ і, при цьому, вони використовують незахищений канал зв'язку, який може прослуховуватися зловмисником (передбачається, що зловмисник не може змінювати дані в ході обміну).
- Аліса та Боб обирають два простих числа g та p, які не є секретними (публічні ключі).
- Аліса генерує випадкове число a, яке буде її приватним ключем, а Боб – b, яке буде його приватним ключем.
- Далі Аліса обчислює проміжний ключ A (2.1):

$$A = g^a \ mod \ p \tag{2.1}$$

- Аліса надсилає бобу отриманий проміжний ключ A, а Боб тим часом обчислює проміжний ключ B за наступною формулою (2.2):

$$B = g^b \ mod \ p \tag{2.2}$$

- Боб надсилає обчислений проміжний ключ В Алісі.
- Аліса, маючи приватний ключ a та проміжний ключ Боба B, обчислює секретний ключ K (2.3):

$$K = B^a \ mod \ p = (g^b \ mod \ p)^a \ mod \ p = g^{ab} \ mod \ p \quad (2.3)$$

- Боб, маючи власний приватний ключ b та проміжний ключ Аліси A, обчислює секретний ключ за наступною формулою (2.4):

$$K = A^b \ mod \ p = (g^a \ mod \ p)^b \ mod \ p = g^{ab} \ mod \ p \quad (2.4)$$

Як бачимо, в ході обчислень, Аліса та Боб отримали спільний секретний ключ K, який може бути використаний в симетричних алгоритмах шифрування з метою захисту інформації, яка передається по незахищеному каналу зв'язку. Схему обміну секретними ключами між Алісою та Бобом зображено на рисунку 2.2.

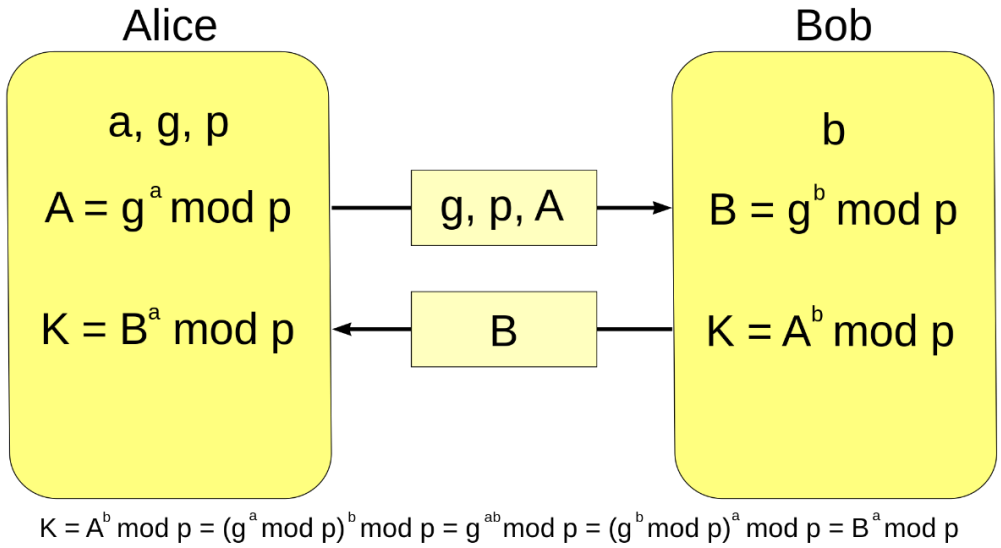


Рисунок 2.2 Обмін секретними ключами за допомогою криптографічного протоколу Діффі-Геллмана

2.3 Модель шифру Цезаря

Шифр Цезаря – це симетричний моноалфавітний шифр підстановки. Даний шифр ще називають шифром зсуву, так як саме зсув лежить в основі даного алгоритму шифрування. Шифрування та розшифрування інформації у шифрі Цезаря відбувається за допомогою заміни відкритого символу на зашифрований символ,

який отримується за допомогою зсуву відкритого символу на певну кількість позицій, яка дорівнює значенню ключа, в межах визначеного алфавіту.

Шифрування даних за допомогою шифру Цезаря описується наступною формулою (2.5):

$$C = (M + K)\ mod\ n, \tag{2.5}$$

де C – зашифрований символ,

M – символ відкритого тексту,

K – ключ шифрування (розшифрування).

n – кількість символів визначеного алфавіту (для англійського алфавіту n = 26)

Таким чином, шифрування тексту відбувається посимвольно і, як результат, на виході утворюється зашифрований текст, кожний символ якого зсунутий на певну кількість позицій в межах визначеного алфавіту, яка відповідає значенню ключа.

Розшифрування даних є оберненим процесом до шифрування і описується наступною формулою:

$$M = (C - K)\ mod\ n \tag{2.6}$$

2.4 Редактор коду Visual Studio Code

Більшість свого часу розробники програмного забезпечення проводять саме у редакторі коду. Редактор коду є одним з основних інструментів розробників, який надає широкий спектр можливостей, які полегшують їх роботу, тим самим збільшуючи їх продуктивність. Зараз існує велика кількість редакторів коду, які мають свої переваги та недоліки.

Загалом редактори коду поділяють на IDE та легкі редактори. IDE є більш потужними редакторами, які надають більший спектр можливостей, ніж легкі редактори. Легкі редактори не мають такої потужності, як IDE, але працюють швидше, мають зручний та простий інтерфейс. На практиці, межі між IDE та

легкими редакторами є дуже розмитими, так як легкі редактори можуть мати велику кількість плагінів, які забезпечують функціонал IDE. До популярних редакторів коду (IDE та легкі редактори) можна віднести:

- Visual Studio
- Visual Studio Code
- Sublime Text
- WebStorm
- Notepad++
- Atom тощо.

Visual Studio Code – це легкий та потужний редактор коду, створений компанією Microsoft, доступний для Windows, macOS та Linux. Він поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов програмування (C++, C#, Go, PHP, Java, Python) і середовищ виконання (.NET і Unity).

Visual Studio Code є гнучким редактором коду, який дозволяє кожному розробнику налаштовувати його під себе. Завдяки підтримці багатьох мов програмування даний редактор дозволяє підвищити продуктивність за допомогою підсвічування синтаксису, автоматичного відступу тощо.

Visual Studio Code включає в себе інтерактивний відладчик коду, який є одним із основних інструментів кожного розробника програмного забезпечення, який дозволяє покроково передивлятися вихідний код, перевіряти змінні, проглядати стеки викликів та виконувати команди в консолі.

Основними перевагами Visual Studio Code є:

- наявність інтегрованих інструментів, які дозволяють проводити тестування коду;
- можливість інтеграції з GitHub;
- широкий вибір можливих тем оформлення;
- наявність функції автодоповнення тощо.

Вікно редактору Visual Studio Code зображено на рисунку 2.3.

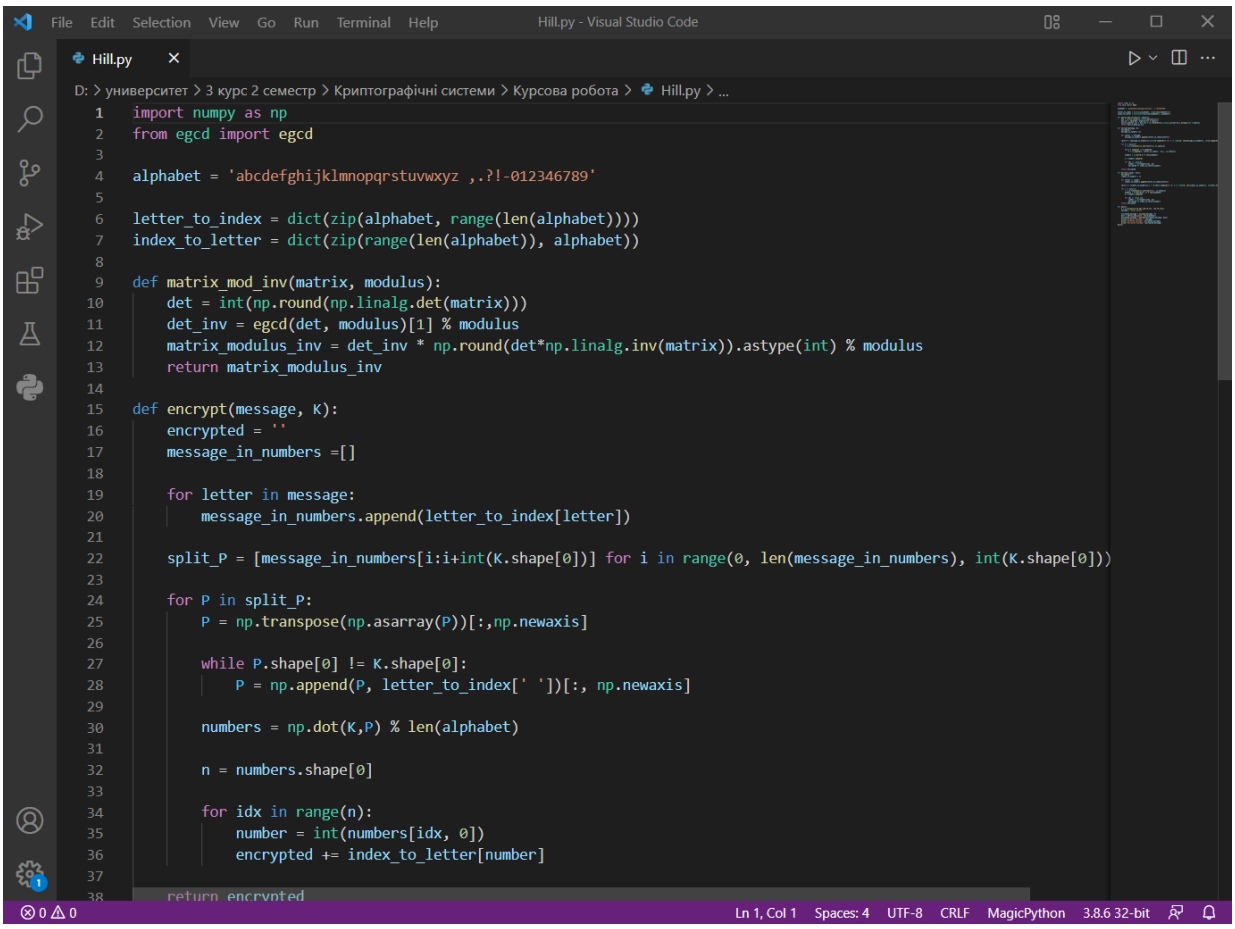


Рисунок 2.3 Visual Studio Code

Редактор коду Visual Studio Code є дуже простим та зручним у використанні, що обумовлює його популярність серед розробників. Так як Visual Studio Code є open-source проектом, він має постійну підтримку та постійно вдосконалюється.

2.5 Мова програмування Python

Python – високорівнева інтерпретована мова програмування, створена голландським програмістом Гвідо Ван Россумом. Python використовує строгу динамічну типізацію. Назва мови пішла від британського серіалу «Monty Python's Flying Circus», фанатом якого був Гвідо.

Python є дуже популярною мовою програмування, яка зараз активно розвивається та використовується в різних сферах. До них можна віднести: штучний інтелект, Big Data, Data Science, ігрова індустрія, мобільні додатки тощо.

До переваг Python можна віднести:

- так як Python є інтерпретованою мовою програмування, програми, написані цією мовою, не потребують компіляції;
- платформонезалежність. Програми, написані мовою Python, можна створювати та запускати на різних операційних системах (Linux, Windows, OS X);
- це open-source проект;
- Python є динамічною мовою програмування, що полегшує написання нескладних програм;
- простота.

До недоліків Python можна віднести:

- низька швидкість виконання порівняно з такими мовами, як C та C++;
- динамічна типізація мови є недоліком при написанні складних програм.

Мова Python дозволяє вирішувати широкий спектр задач. Їх можна розбити на наступні категорії:

1. Програми баз даних. У Python є інтерфейси доступу до всіх основних реляційних баз даних: Sybase, Oracle, Informix, ODBC, MySQL, PostgreSQL, SQLite та багатьох інших.

2. Веб-програми. За допомогою додаткових фреймворків (Django, Flask, Pyramid) можна створювати повнофункціональні сайти.

3. Веб-сценарії. Python поставляється разом зі стандартними інтернет-модулями, які дозволяють програмам виконувати різноманітні мережеві операції як у режимі клієнта, так і в режимі сервера.

4. Системне програмування. Вбудовані в Python інтерфейси доступу до служб операційних систем роблять його ідеальним інструментом для створення утиліт системного адміністрування.

Проекти, в яких використовується Python:

- Сервіс YouTube значною мірою реалізований на Python.
- Агенство національної безпеки (NSA) використовує Python для шифрування і аналізу даних.

- NASA, Los Alamos, JPL і Fermilab використовують Python для наукових обчислень.
- Компанія Google використовує Python у власній пошуковій системі.
- Компанії Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm і IBM використовують Python для тестування апаратного забезпечення.

2.6 Технології віртуалізації для тестування програмного забезпечення

Наступним етапом після написання коду є тестування створеного програмного забезпечення на реальних обчислювальних системах. В більшості випадків розробники тестують працездатність своїх програм на віртуальних машинах. Віртуальна машина – це спеціальне програмне забезпечення, яке емулює роботу фізичної машини [7].

Для створення та запуску віртуальних машин використовують спеціальне програмне забезпечення. Серед програм для роботи з віртуальними машинами можна виділити VMware Workstation та Oracle VM VirtualBox. Вони є доволі популярними середовищами віртуалізації, які надають багато можливостей та мають свої переваги та недоліки.

Oracle VM VirtualBox є безкоштовним open-source проектом, на відміну від частково безкоштовного VMware Workstation. Перевагою VMware Workstation над Oracle VM VirtualBox при встановленні віртуальних машин є те, що VMware Workstation має велику кількість шаблонів для різних операційних систем, що дозволяє встановлювати їх автоматично, налаштувавши лише основні параметри. Щоб встановити віртуальну машину на Oracle VM VirtualBox необхідно вказати велику кількість налаштувань і весь процес встановлення проходить вручну.

Вікна Oracle VM VirtualBox та VMware Workstation зображені на рисунках 2.4 та 2.5 відповідно.

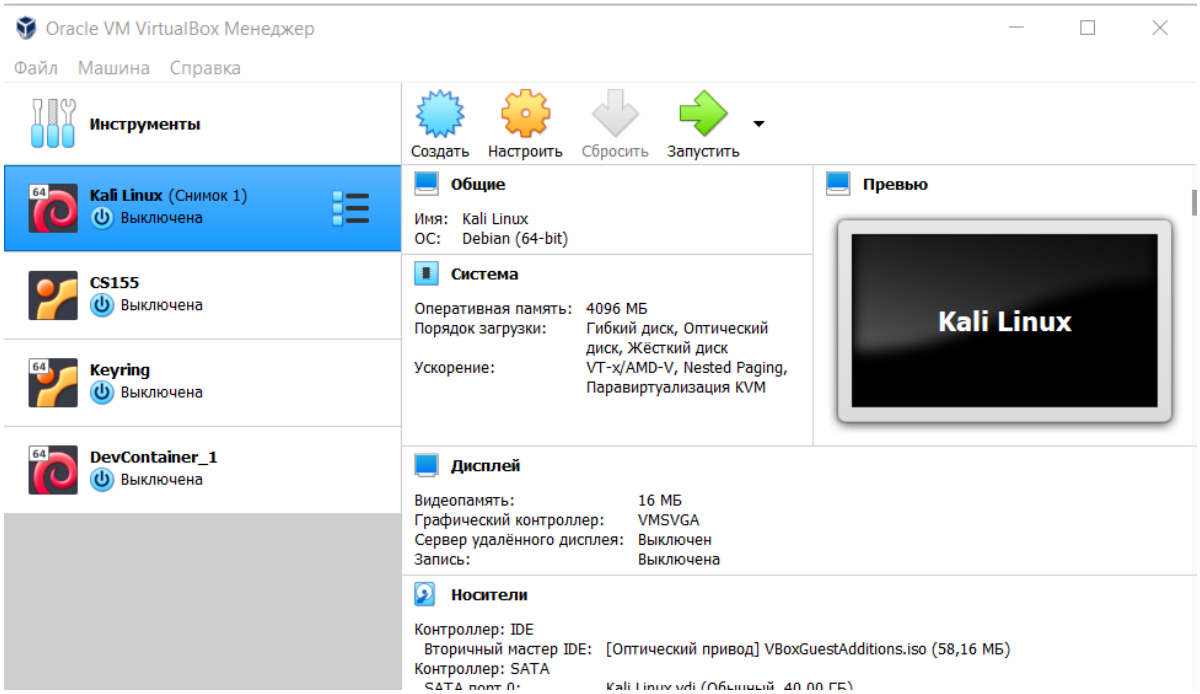


Рисунок 2.4 Oracle VM VirtualBox

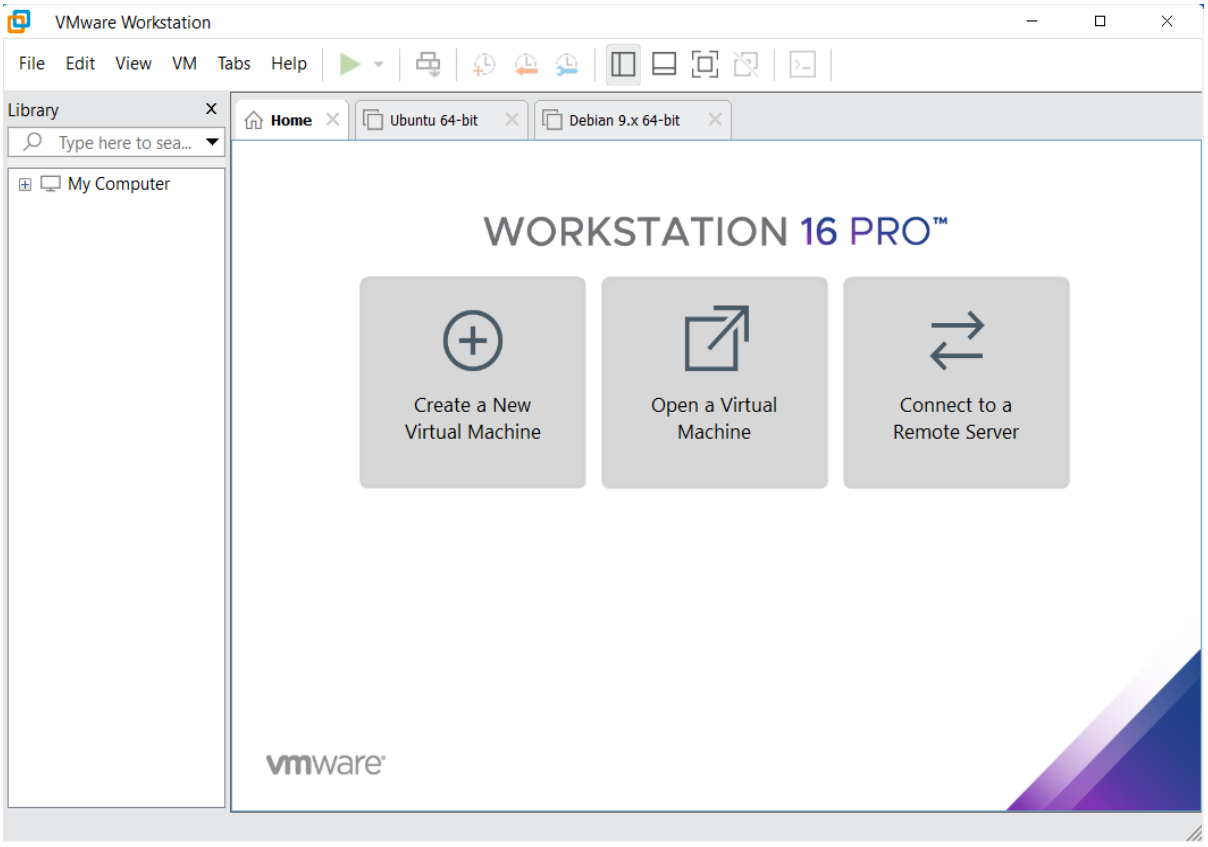


Рисунок 2.5 VMware Workstation

Обидва продукти є потужними та мають великий спектр можливостей. Вибір програмного забезпечення для віртуалізації залежить лише від особистих потреб та вподобань.

Висновки за розділом 2

В даному розділі визначено архітектуру програмного забезпечення, яка є своєрідним шаблоном за яким створюється програмне забезпечення. Також описано основні компоненти, які є необхідними для створення програмного забезпечення:

- моделі використаних алгоритмів – шифру Цезаря та криптографічного протоколу Діффі-Геллмана;
- редактор коду Visual Studio Code, який надає багато можливостей для створення програмного забезпечення і спрощує процес написання коду;
- мову програмування Python, яка використовувалась для створення програмного забезпечення;
- технології віртуалізації для тестування програмного забезпечення.

‘

РОДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБМІНУ СЕКРЕТНИМИ КЛЮЧАМИ НА БАЗІ ПРОТОКОЛУ ДІФФІ-ГЕЛЛМАНА

3.1 Загальний опис та структура програмного забезпечення

Програмне забезпечення написано мовою Python. Дане програмне забезпечення працює на наступних ОС: Windows та Linux. Створене ПЗ реалізує механізм обміну секретними ключами між клієнтом та сервером з використанням криптографічного протоколу Діффі-Геллмана, з метою подальшого їх використання в симетричному алгоритмі шифрування – шифрі Цезаря. В якості прикладу практичного застосування криптографічного протоколу Діффі-Геллмана було обрано процес проходження авторизації клієнта на сервері.

Структурно програмне забезпечення складається з трьох фалів:

- server.py;
- client.py;
- functions.py.

Файл server.py відповідає за роботу сервера. Даний файл реалізовує запуск сервера, під’єднання клієнтів та обробку клієнтських запитів. Даний файл також реалізує процес проходження ідентифікації та автентифікації і, як наслідок, авторизації.

Файл client.py містить в собі логіку роботи клієнта. Даний файл реалізує механізм під’єднання клієнта до сервера, надсилання запитів від клієнта до сервера, отримання та обробку відповідей, наданих сервером.

Файл functions.py включає в себе класи та методи, які використовуються в файлах client.py та server.py для забезпечення наступних процесів:

- обміну секретними ключами за допомогою криптографічного протоколу Діффі-Геллмана;

- шифрування та розшифрування інформації з використанням шифру Цезаря.

Програмне забезпечення забезпечує шифрування даних між клієнтом та сервером з використанням криптографічного протоколу Діффі-Геллмана та шифру Цезаря. Шифрування даних під час під час обміну ними забезпечує збереження однієї з основних властивосте інформації – конфіденційності.

3.2 Опис використаних модулів

При створенні програмного забезпечення були використані наступні бібліотеки Python:

- socket;
- random.

Модуль socket

Модуль socket в Python використовується для роботи з сокетами. Сокет - це програмний інтерфейс задля забезпечення інформаційного обміну між процесами. Використання даного модулю забезпечує можливість обміну даними між клієнтом та сервером. Є два види сокетів: серверний та клієнтський. Серверний сокет прослуховує певний порт, а клієнтський підключається до сервера. Після успішного підключення починається обмін даними між клієнтом та сервером. Сокети працюють на транспортному рівні моделі OSI та бувають двох типів: потокові (на базі протоколу TCP) та дейтаграмні (на базі UDP протоколу).

Для роботи з сокетами було використано наступні функції модуля socket:

- socket() – створює новий сокет;
- bind() – прив’язує сокет до адреси;
- listen() – дозволяє серверу приймати з’єднання;
- connect() – підключає до віддаленого сокета за адресою;
- accept() - приймає з’єднання;
- send() – надсилає дані в сокет;
- recv() – отримує дані з сокета;
- shutdown() – вимикає одну або обидві половини з’єднання;

- close() – закриває файловий дескриптор сокета.

Модуль random

Модуль random використовується для генерації псевдовипадкових чисел, вибору випадкового елемента в послідовності. В основі даного модуля лежить генератор псевдовипадкових чисел Mersenne Twister, який є одним з генераторів, які пройшли найретельніше тестування.

В створеному ПЗ було використано наступні функції модуля random:

- randint() – генерує випадкове ціле число з заданого проміжку;
- choice() – обирає випадковий елемент з непустої послідовності.

3.3 Функції та методи для роботи з базовими типами

Для створення ПЗ було використано функції для роботи з базовими типами даних. До базових типів мови Python відносять: рядок, число, список, кортеж, словник тощо.

Для роботи з рядками було використано наступні функції:

- encode(). Дана функція використовується для кодування рядку (перетворення рядку в байти);
- decode(). Функція decode() є протилежною до функції encode(). Вона перетворює байти в об’єкт з типом string;
- find(). Дана функція використовується для перевірки входження підрядка в рядок;
- split(). Використовується для розділення рядку за певним символом.

Для роботи зі словниками було використано функцію get(), за допомогою якої можна отримати значення ключа переданого в дану функцію.

Для роботи зі списками було використано функцію append(), яка дозволяє додавати елементи в список.

3.4 Опис класів та функцій

Основні класи та функції, використані в створеному ПЗ, містяться в файлі functions.py. В даному файлі знаходяться класи та функції, які використовуються для вирішення наступних завдань:

- обмін секретними ключами за допомогою криптографічного протоколу Діффі-Геллмана;
- шифрування та розшифрування повідомлень за допомогою шифру Цезаря;
- ідентифікація, автентифікація та авторизація.

Обмін секретними ключами за допомогою криптографічного протоколу Діффі-Геллмана

Для реалізації процесу обміну ключами було створено клас Diffie_Hallman. Сюди входять наступні методи:

- __init__();
- public_keys();
- first();
- res_key();
- exchange_client();
- exchange_server().

Метод __init__()

Метод __init__() є конструктором класу, який автоматично викликається при створення об’єкта. Даний метод має наступні параметри:

- self - вказує на конкретний екземпляр класу;
- public_key_1, public_key_2 – прості числа, які обираються абонентами перед обміном і не є секретними.
- private_key – секретне число, яке обирає кожен абонент і тримає його в секретності.

В даному методі створюються змінні, яким привласнюються передані в метод аргументи з метою їх подальшого використання в інших методах даного класу.

Метод public_keys()

Даний метод не має ніяких параметрів і повертає кортеж, який складається з двох простих чисел, які будуть використовуватися в процесі обміну ключами. В даному методі відбувається створення списку з простих чисел, з якого потім обирається два різних числа (перше число менше другого) і повертаються у вигляді кортежу. Схема роботи даного методу наведена на рисунку 3.1.

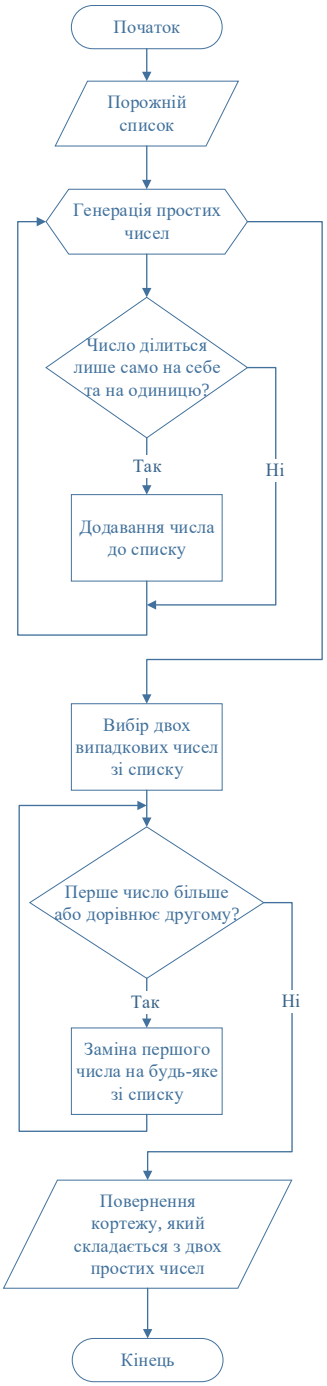


Рисунок 3.1 Схема роботи методу public_keys()

Метод first()

Даний метод приймає в якості параметрів два публічних ключа та один приватний ключ. За допомогою цих ключів виконується обчислення проміжного ключа, який повертається з цього методу. Схема роботи методу first() зображена на рисунку 3.2.

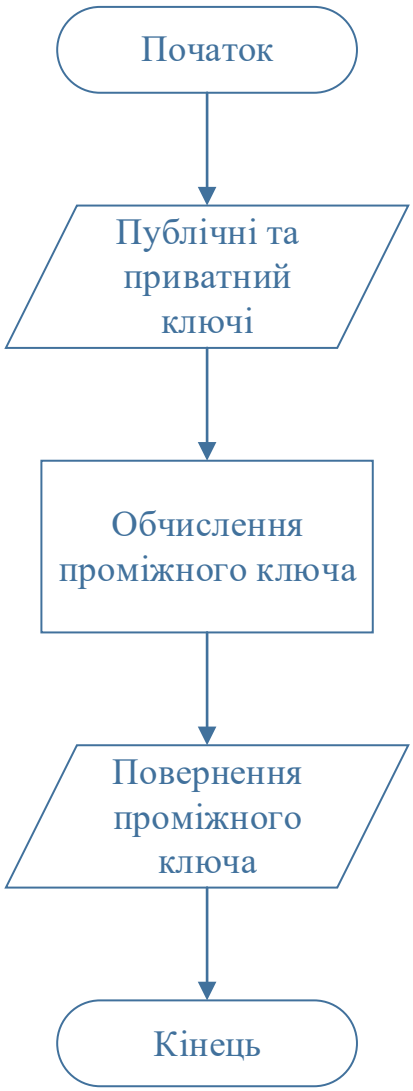


Рисунок 3.2 Схема роботи методу first()

Метод res_key()

Роль даного методу полягає в обчисленні результуючого ключа, який буде однаковим як для клієнта, так і для сервера. На вхід даному методу подається проміжний ключ, з використанням якого обчислюється результуючий секретний ключ. Схематично роботу даного методу наведено на рисунку 3.3.

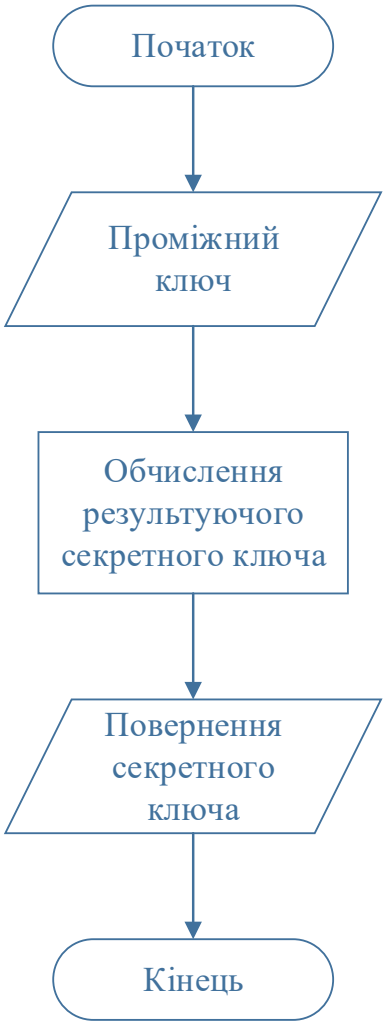


Рисунок 3.3 Схема роботи методу res_key()

Метод exchange_client()

Даний метод використовується на стороні клієнта і забезпечує процес обміну даними між клієнтом та сервером з метою обчислення секретного ключа з використанням методів класу Diffie_Hellman. На вхід даному методу подається клієнтський сокет. Клієнт отримує від сервера публічні ключі. Далі виконується генерація випадкового числа, яке слугує приватним ключем. Після генерації приватного ключа за допомогою методів класу обраховується проміжний ключ, який надсилається серверу. Потім, після отримання проміжного ключа від сервера, виконується обчислення секретного ключа за допомогою методу res_key(). Схему роботи даного методу наведено на рисунку 3.4.

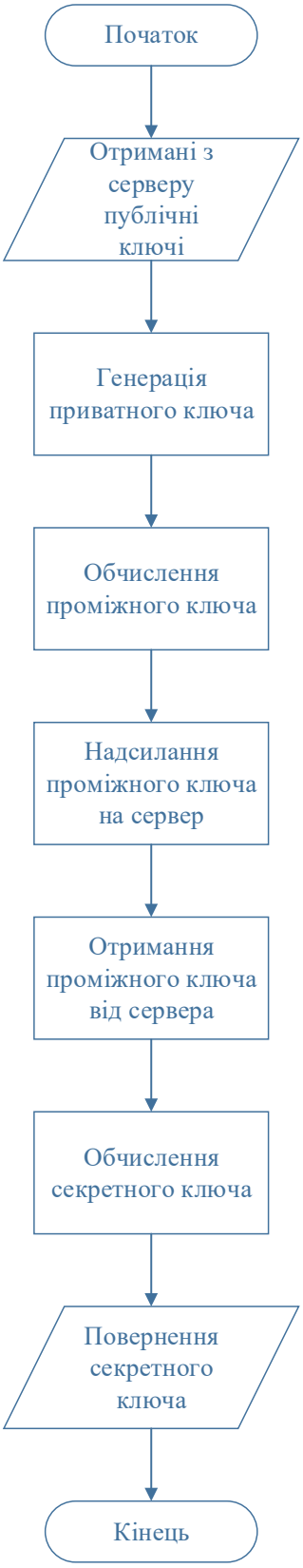
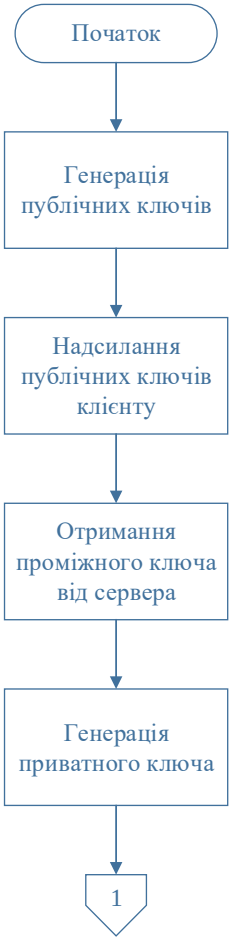


Рисунок 3.4 Схема роботи методу exchange_client()

Метод exchange_server()

Даний метод використовується на стороні сервера для забезпечення обміну даними з клієнтом з метою обчислення секретного ключа за допомогою

криптографічного протоколу Діффі-Геллмана. На вхід подається серверний сокет. Всередині цього методу відбувається генерація публічних ключів на сервері, які відправляються клієнту. Наступним кроком є отримання проміжного ключа, обчисленого на стороні клієнта. Потім відбувається генерація приватного ключа для сервера. За допомогою публічних та приватного ключа сервер обчислює проміжний ключ. Після обчислення проміжного ключа відбувається порівняння проміжного ключа сервера та проміжного ключа клієнта. Якщо вони однакові, то сервер генерує новий приватний ключ та обчислює новий проміжний ключ до того моменту, поки проміжний ключ клієнта та проміжний ключ сервера не будуть різними. Потім, обчислений сервером, проміжний ключ надсилається клієнту. Далі обчислюється секретний ключ, який повертається з методу. Схематично роботу даного методу зображено на рисунку 3.5.



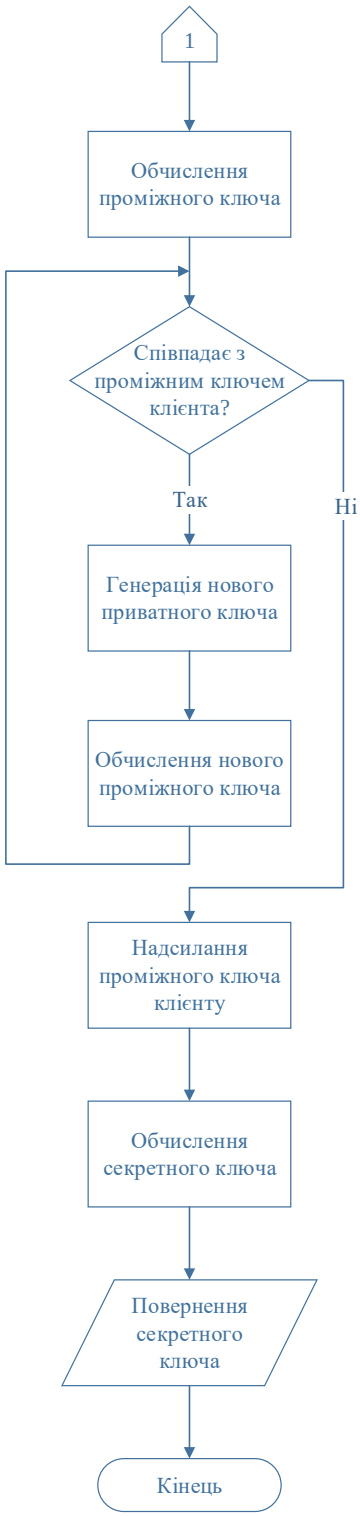


Рисунок 3.5 Схема роботи методу exchange_server()

Шифрування та розшифрування повідомлень за допомогою шифру Цезаря

Після обчислення секретного ключа клієнтом та сервером він використовується в симетричному алгоритмі шифрування (шифрі Цезаря) для шифрування та розшифрування даних, якими обмінюються клієнт та сервер.

Для реалізації шифрування та розшифрування інформації шифром Цезаря було створено клас Caesar. Він включає в себе наступні поля:

- al_up – рядок, який складається з символів англійського алфавіту у верхньому регістрі;
- al_low – рядок, який складається з символів англійського алфавіту у нижньому регістрі;
- symbols – рядок, який складається з спеціальних символів та чисел.

Для шифрування та розшифрування інформації за допомогою даного шифру було створено два методи:

- encrypt()
- decrypt()

Метод encrypt()

Метод encrypt() використовується для шифрування даних за допомогою шифру Цезаря. На вхід подається відкритий текст та ключ. В даному методі створюється змінна, яка буде містити в собі зашифрований текст. На самому початку в цій змінній знаходиться пустий рядок, в який потім, в ході шифрування, додаються зашифровані символи. Шифрування відбувається наступним чином:

1. За допомогою циклу відбувається проходження по кожному символу відкритого тексту (кожний символ шифрується окремо).

2. Виконується перевірка входження символів в рядки al_up, al_low та symbols.

3. Якщо символ знаходиться в одному з рядків (al_up, al_low або symbols), то відбувається заміна даного символу на інший, який отримується за допомогою зсуву відкритого символу на значення ключа в межах рядку в якому він був знайдений (al_up, al_low або symbols). Далі зашифрований символ записується (додається) в змінну, в якій буде зберігатися зашифрований текст.

4. Якщо символ не знаходиться в жодному з рядків (al_up, al_low або symbols), то він не змінюється та записується (додається) в змінну, в якій буде зберігатися зашифрований текст.

Після завершення шифрування, на виході отримуємо зашифрований текст, який повертається з даного методу. Схему роботи даного методу зображено на рисунку 3.6.

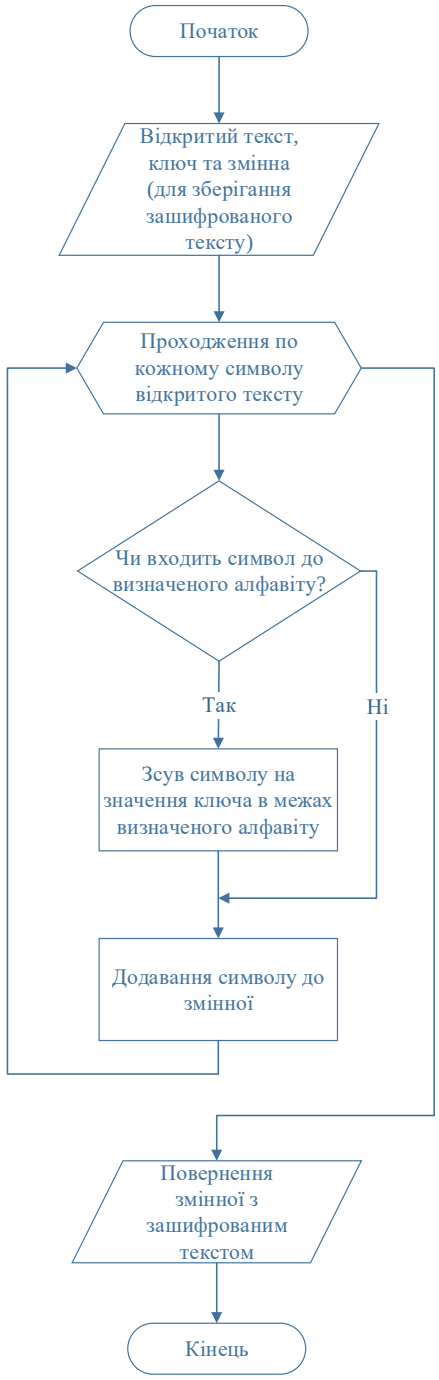


Рисунок 3.6 Схема роботи методу encrypt()

Метод decrypt()

Метод encrypt() реалізує функцію розшифрування тексту за допомогою шифру Цезаря. На вхід подається зашифрований текст та ключ. Перед початком процесу розшифрування створюється змінна, яка буде містити в собі розшифрований текст. Процес розшифрування відбувається так само як і процес шифрування, тільки зсув символів відбувається в протилежному напрямку. Після завершення розшифрування, на виході отримуємо змінну в якій знаходиться відкритий (розшифрований) текст. Дана змінна повертається з даного методу. Схему роботи методу decrypt() зображено на рисунку 3.7.

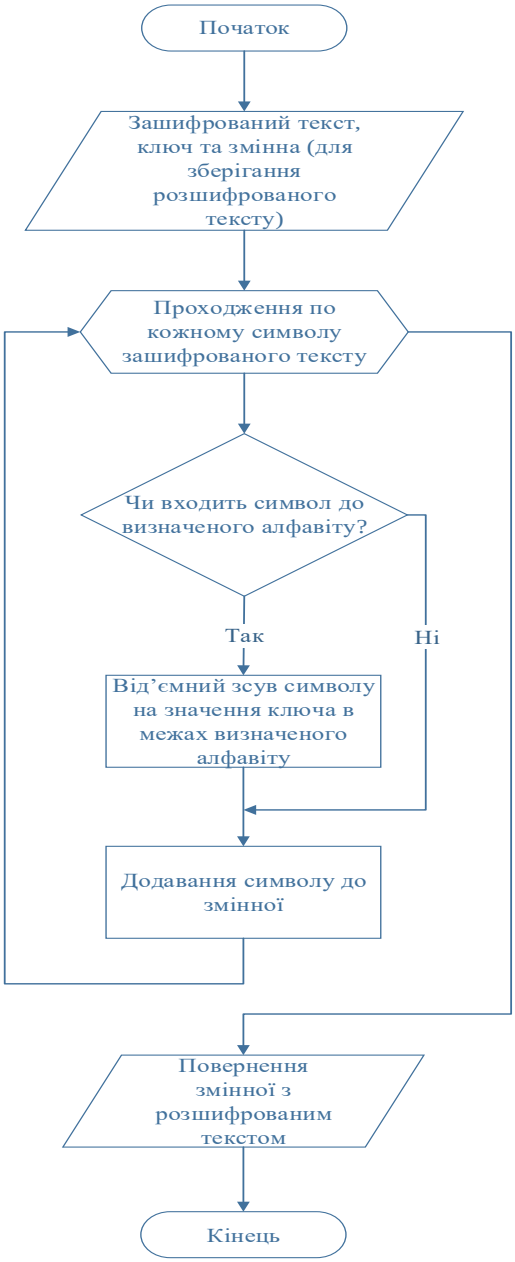


Рисунок 3.7 Схема роботи методу decrypt()

Ідентифікація, автентифікація та авторизація

Частково процес проходження ідентифікації, автентифікації та авторизації описаний у файлі functions.py у вигляді двох функції:

- identification()
- authentication()

Функція identification()

На вхід даній функції подається словник, в якому містяться логіни зареєстрованих користувачів та їх паролі, та логін, який вводить користувач. Дана функція перевіряє входження введеного логіну до словника, порівнюючи введений логін з усіма наявними в словнику логінами за допомогою циклу. Якщо введений логін входить до словника, то функція повертає логічне значення true, якщо ні – логічне значення false. Схема роботи даної функції наведена на рисунку 3.8.

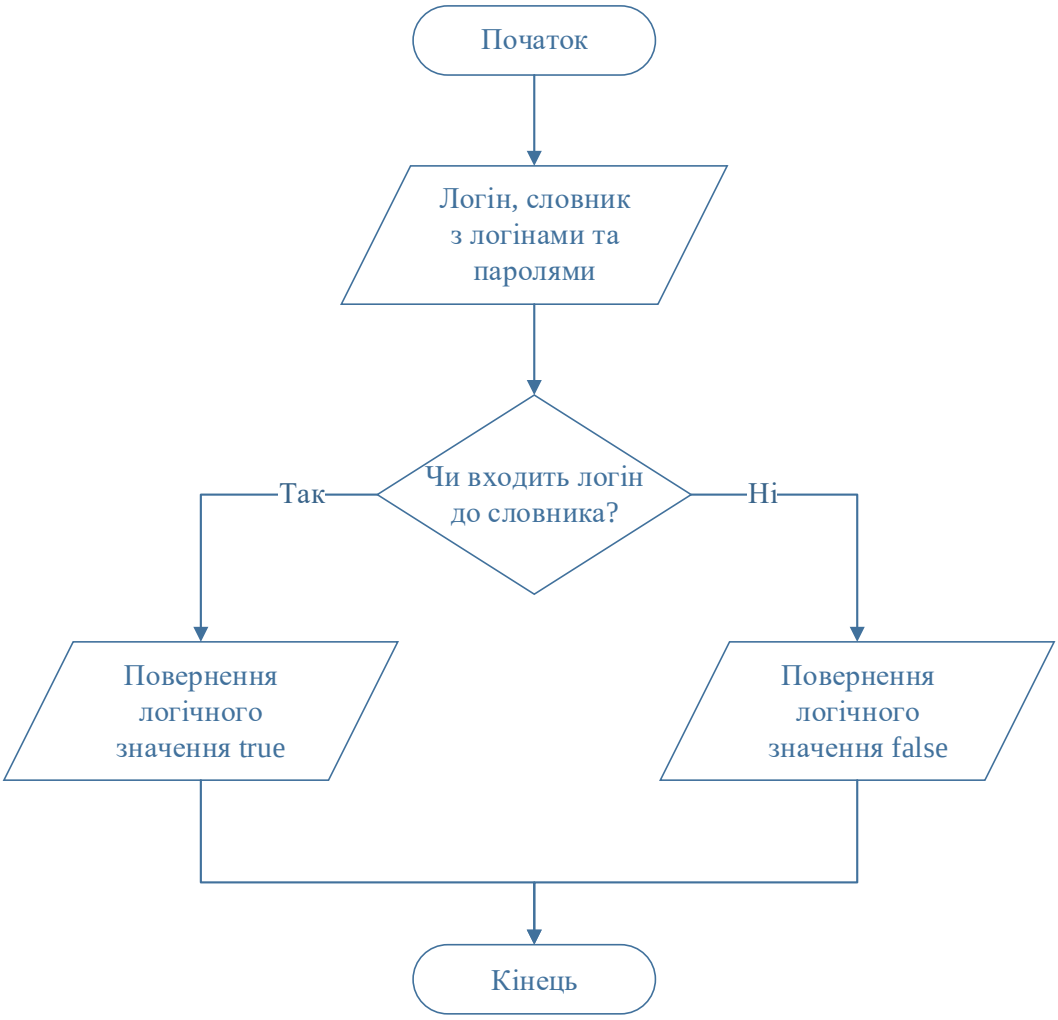


Рисунок 3.8 Схема роботи функції identification()

Функція authentication()

Дана функція приймає три параметри:

- логін та пароль введені користувачем;
- словник з логінами та паролями користувачів.

Дана функція виконується лише при умові того, що функція identification() повернула логічне значення true, що вказує на те, що введений логін входить до словника. Завдання даної функції полягає в визначенні відповідності між введеним користувачем паролем та паролем, який знаходиться в словнику і відповідає введеному раніше логіну (який вже пройшов перевірку входження до словнику). Якщо введений пароль співпадає з паролем в словнику, то повертається логічне значення true, якщо ні – логічне значення false. Схема роботи даної функції зображена на рисунку 3.9.

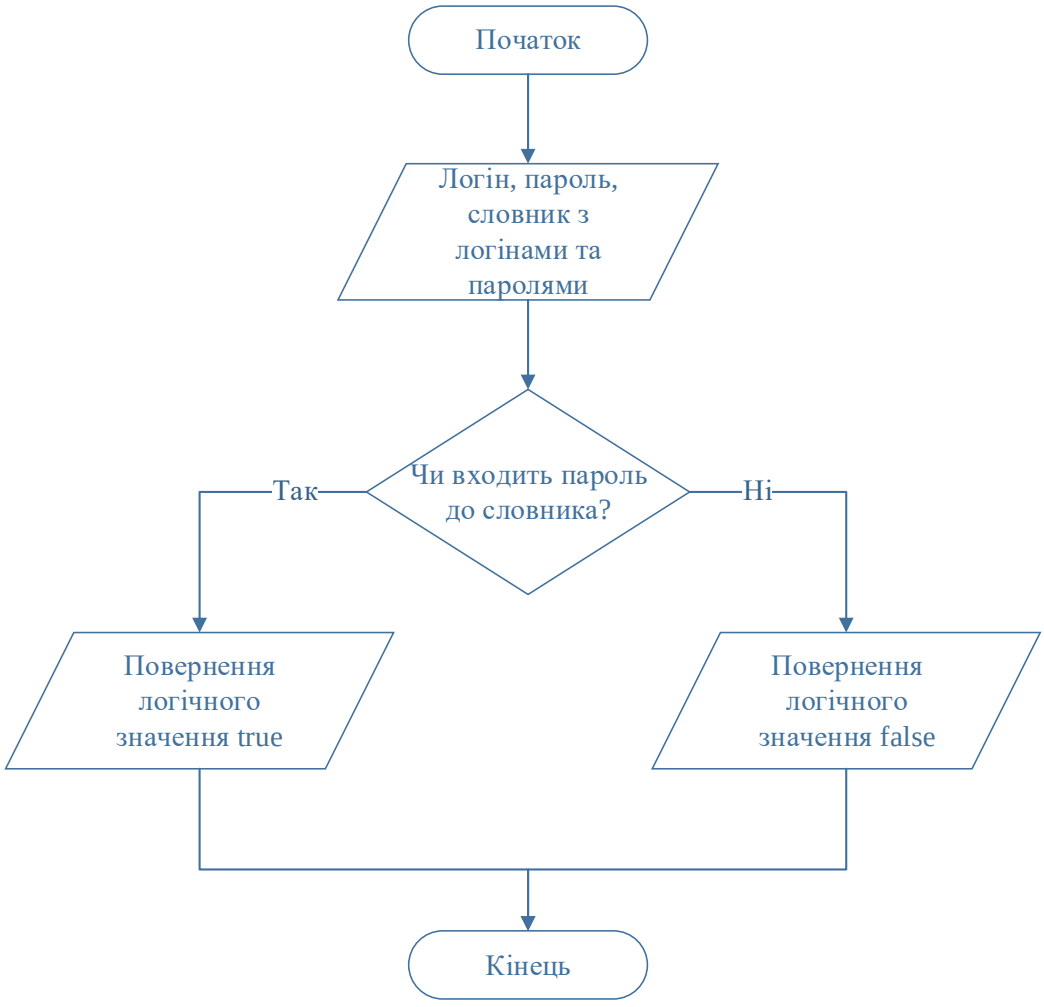


Рисунок 3.9 Схема роботи функції authentication

3.5 Опис роботи сервера та клієнта

Робота сервера та клієнта описана в файлах server.py та client.py відповідно. В даних файлах реалізується процес обміну даними між клієнтом та сервером з використанням класів, методів та функцій, описаних в файлі functions.py.

Файл server.py

Файл server.py відповідає за запуск та роботу сервера. В даному файлі використовуються функцій, класи та методи, які описані в файлі functions.py. Файл server.py відповідає за виконання наступних процесів:

- Запуск сервера (створення серверного сокета, привласнення йому адреси та порту).
- Обмін даними між сервером та клієнтом (надсилання, приймання та обробка даних).
- Обмін ключами за допомогою криптографічного протоколу Діффі-Геллмана. Для реалізації даного процесу використовується клас Diffie-Hellman та його методи, описані в файлі functions.py.
- Розшифрування інформації, що отримується від клієнта, за допомогою шифру Цезаря. Для реалізації даного процесу використовується клас Caesar та його методи, описані в файлі functions.py.
- Ідентифікація, автентифікація та авторизація. Частина процесу реалізовується за допомогою функцій identification() та authentication(), описаних в файлі functions.py. Інша частина реалізовується в самому файлі server.py. В файлі server.py визначена кількість можливих спроб для проходження ідентифікації та автентифікації. В результаті вдалого (невдалого) проходження ідентифікації або автентифікації клієнту надсилається відповідне повідомлення.
- Припинення обміну даними з клієнтським сокетом, завершення з’єднання з клієнтським сокетом.

Схематично процес роботи сервера, описаного у файлі server.py, наведено на рисунку 3.10.

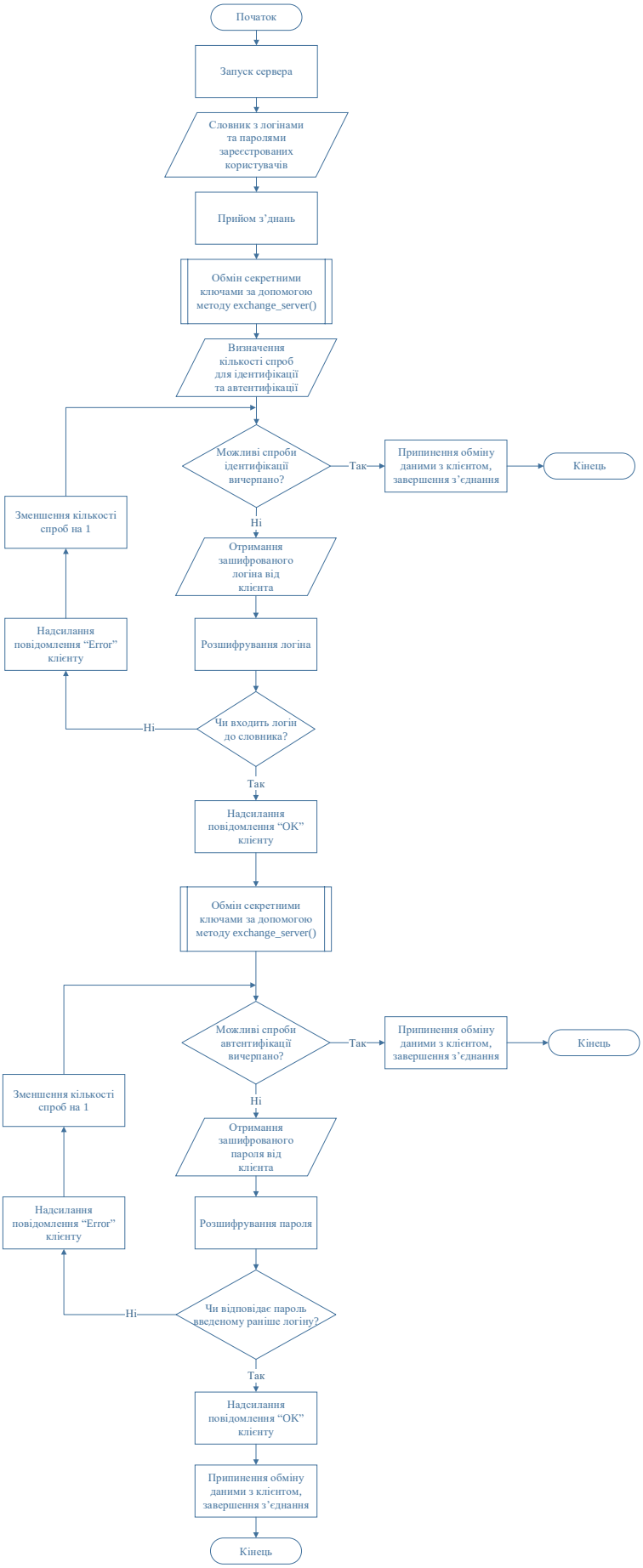


Рисунок 3.10 Схема роботи сервера

Файл client.py

Даний файл відповідає за роботу клієнтської частини. В даному файлі, так як і в файлі server.py, використовуються класи, методи та функції, описані в файлі functions. Файл client.py відповідає за виконання наступних процесів:

- Створення клієнтського сокета, під’єднання до сервера.
- Обмін даними між клієнтом та сервером (надсилання, приймання та обробка даних).
- Обмін ключами за допомогою криптографічного протоколу Діффі-Геллмана. Для реалізації даного процесу використовується клас Diffie-Hellman та його методи, описані в файлі functions.py.
- Шифрування інформації, що передається на сервер, за допомогою шифру Цезаря. Для реалізації даного процесу використовується клас Caesar та його методи, описані в файлі functions.py.
- Ідентифікація, автентифікація та авторизація. Частина процесу реалізовується за допомогою функцій identification() та authentication(), описаних в файлі functions.py. Інша частина реалізовується в самому файлі client.py. В файлі client.py визначена кількість можливих спроб для проходження ідентифікації та автентифікації. В результаті вдалого (невдалого) проходження ідентифікації або автентифікації клієнт отримує від сервера відповідне повідомлення.
- Припинення обміну даними з серверним сокетом, завершення з’єднання з серверним сокетом.

Загалом, механізми роботи сервера (server.py) та клієнта (client.py) схожі між собою та виконують схожі задачі з метою реалізації обміну секретними ключами за допомогою криптографічного протоколу Діффі-Геллмана, які використовуються для шифрування (розшифрування) інформації (ідентифікаційних та автентифікаційних даних) за допомогою шифру Цезаря.

Схему роботи клієнта, описаного у файлі client.py, зображено на рисунку 3.11.

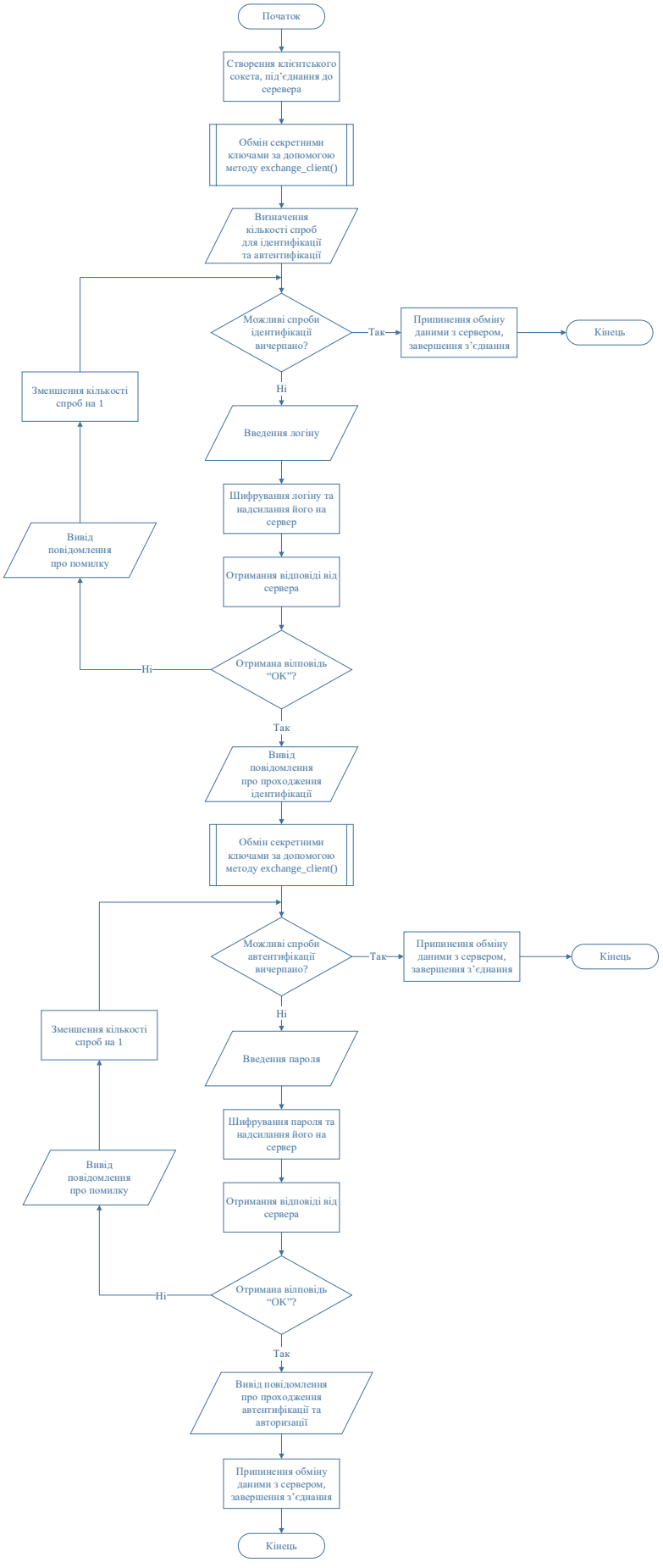


Рисунок 3.11 Схема роботи клієнта

3.6 Тестування програмного забезпечення

Для тестування створеного програмного забезпечення було використано чотири віртуальні машини:

- Віртуальна машина (сервер) з встановленою ОС Kali Linux та IP-адресою 192.168.1.12 (рис. 3.12).

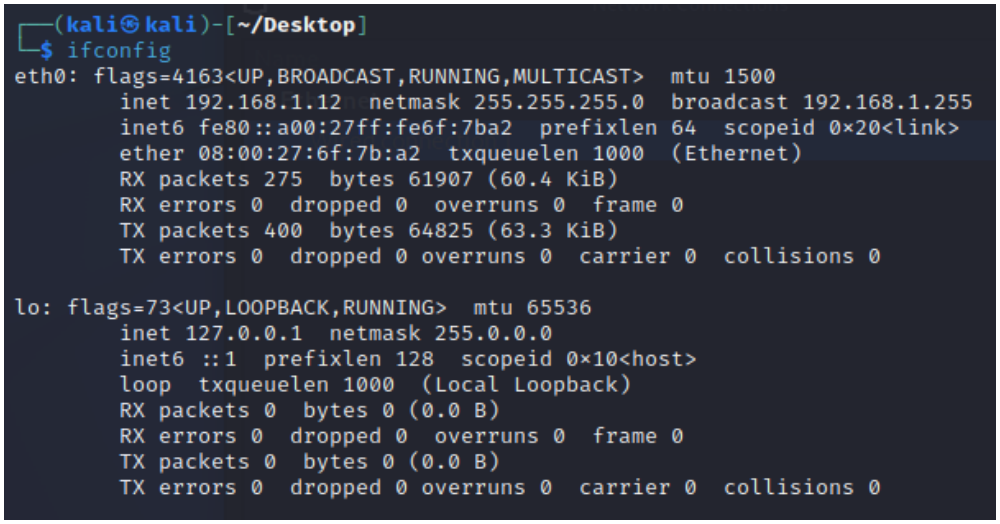


Рисунок 3.12 Віртуальна машина (сервер) з встановленою ОС Kali Linux

- Віртуальна машина (клієнт) з встановленою ОС Kali Linux та IP-адресою 192.168.1.13 (рис. 3.13).

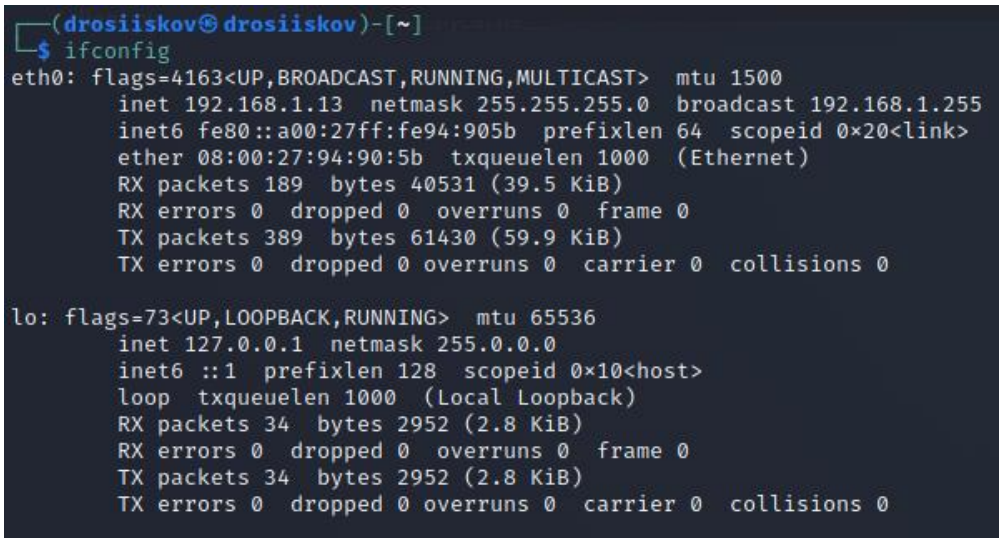


Рисунок 3.13 Віртуальна машина (клієнт) з встановленою ОС Kali Linux

- Віртуальна машина (сервер) з встановленою ОС Windows 10 та IP-адресою 192.168.1.10 (рис. 3.14).

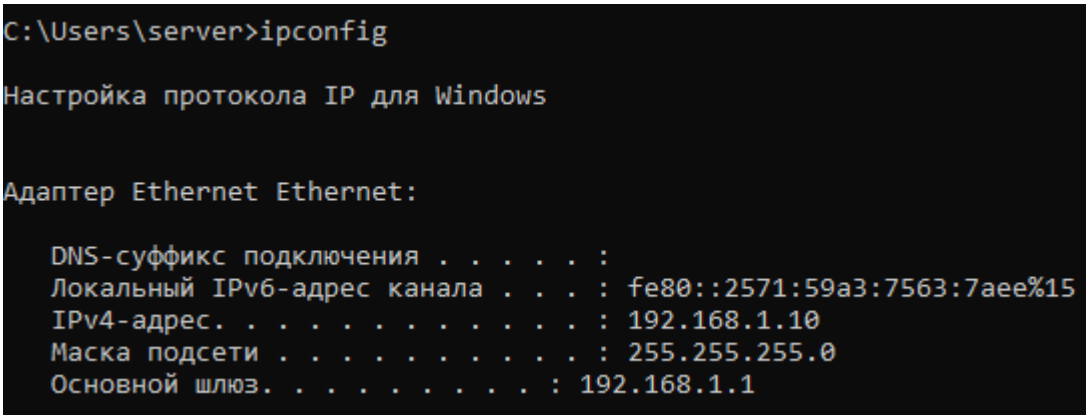


Рисунок 3.14 Віртуальна машина (сервер) з встановленою ОС Windows 10

- Віртуальна машина (клієнт) з встановленою ОС Windows 10 та IP-адресою 192.168.1.11 (рис. 3.15).

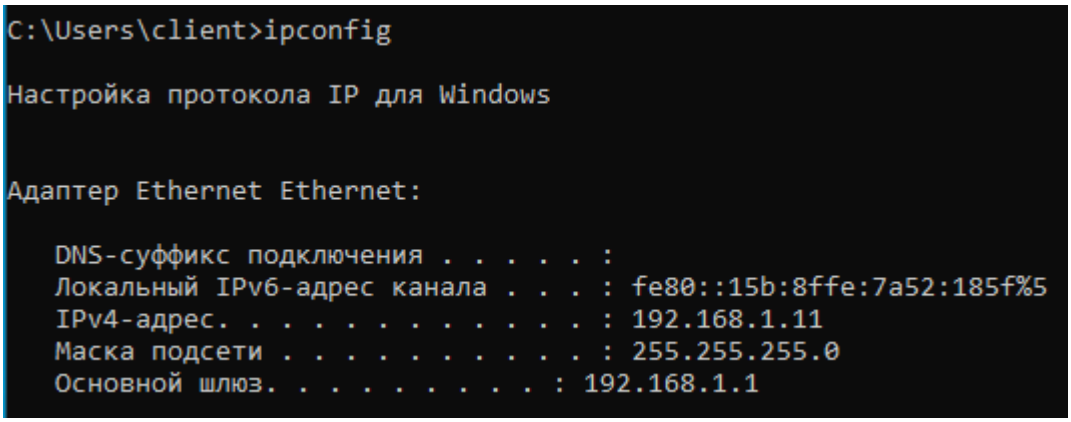


Рисунок 3.15 Віртуальна машина (клієнт) з встановленою ОС Windows 10

На створені для тестування програмного забезпечення віртуальні машини було завантажено наступні файли:

- server.py та functions.py на віртуальні машини з ОС Kali Linux та Windows 10, які будуть відповідати за роботу серверної частини;
- client.py та functions.py на віртуальні машини з ОС Kali Linux та Windows 10, які будуть відповідати за роботу клієнтської частини.

Після встановлення віртуальних машин та необхідних файлів було перевірено працездатність створеного програмного забезпечення. Тестування програмного забезпечення (клієнтської та серверної частини) проводилось як на однакових ОС, так і на різних.

Під час тестування програмного забезпечення було використано наступні комбінації операційних систем:

- Kali Linux – сервер; Kali Linux - клієнт (рис. 3.16 - 3.17).

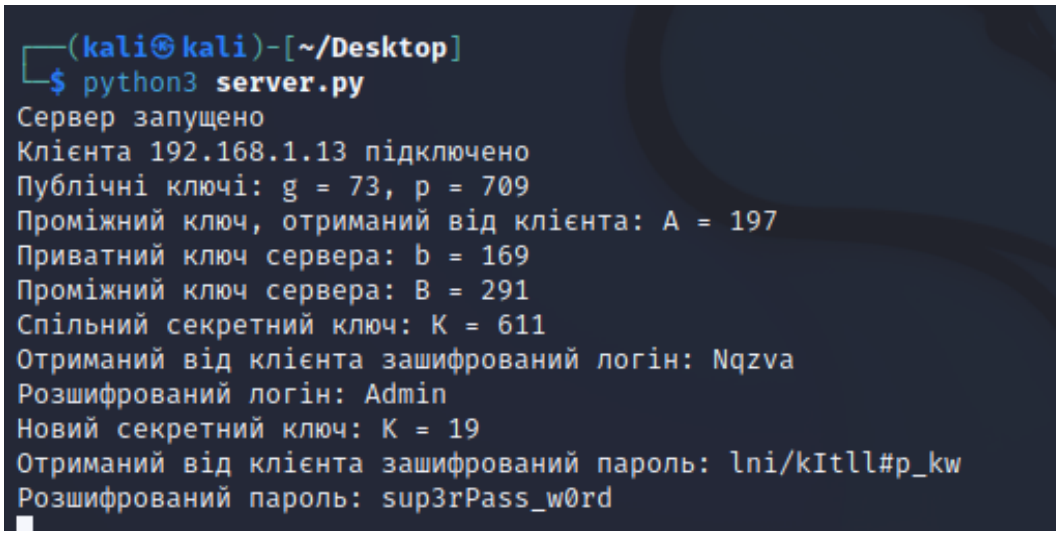


Рисунок 3.16 Робота сервера на Kali Linux

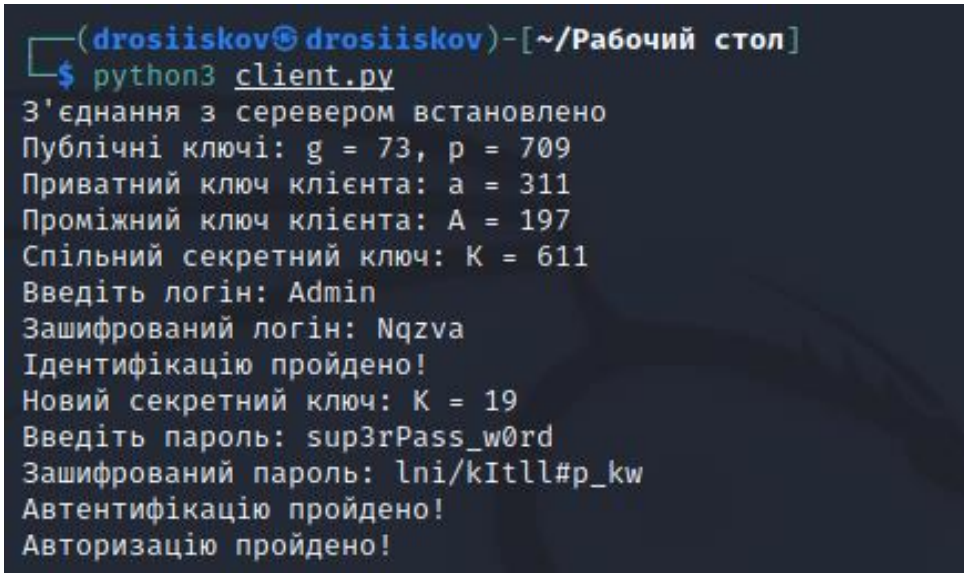


Рисунок 3.17 Робота клієнта на Kali Linux

- Windows 10 – сервер; Windows 10 – клієнт (рис 3.18 – 3.19).

```
C:\Users\server>server.py
Сервер запущено
Клієнта 192.168.1.11 підключено
Публічні ключі: g = 19, p = 73
Проміжний ключ, отриманий від клієнта: A = 67
Приватний ключ сервера: b = 109
Проміжний ключ сервера: B = 19
Спільний секретний ключ: K = 67
Отриманий від клієнта зашифрований логін: Bxrwpta
Розшифрований логін: Michael
Новий секретний ключ: K = 165
Отриманий від клієнта зашифрований пароль: zfnach
Розшифрований пароль: qwerty
```

Рисунок 3.18 Робота сервера на Windows 10

```
C:\Users\client>client.py
З'єднання з серевером встановлено
Публічні ключі: g = 19, p = 73
Приватний ключ клієнта: a = 535
Проміжний ключ клієнта: A = 67
Спільний секретний ключ: K = 67
Введіть логін: Michael
Зашифрований логін: Bxrwpta
Ідентифікацію пройдено!
Новий секретний ключ: K = 165
Введіть пароль: qwerty
Зашифрований пароль: zfnach
Автентифікацію пройдено!
Авторизацію пройдено!
```

Рисунок 3.19 Робота клієнта на Windows 10

- Windows 10 – сервер; Kali Linux – клієнт (рис. 3.20 – 3.21).

```
C:\Users\server>server.py
Сервер запущено
Клієнта 192.168.1.13 підключено
Публічні ключі: g = 859, p = 919
Проміжний ключ, отриманий від клієнта: A = 176
Приватний ключ сервера: b = 501
Проміжний ключ сервера: B = 840
Спільний секретний ключ: K = 301
Отриманий від клієнта зашифрований логін: Ydwc
Розшифрований логін: John
Новий секретний ключ: K = 110
Отриманий від клієнта зашифрований пароль: nkrRu;Cuxrj
Розшифрований пароль: helLo_World!
```

Рисунок 3.20 Робота сервера на Windows 10

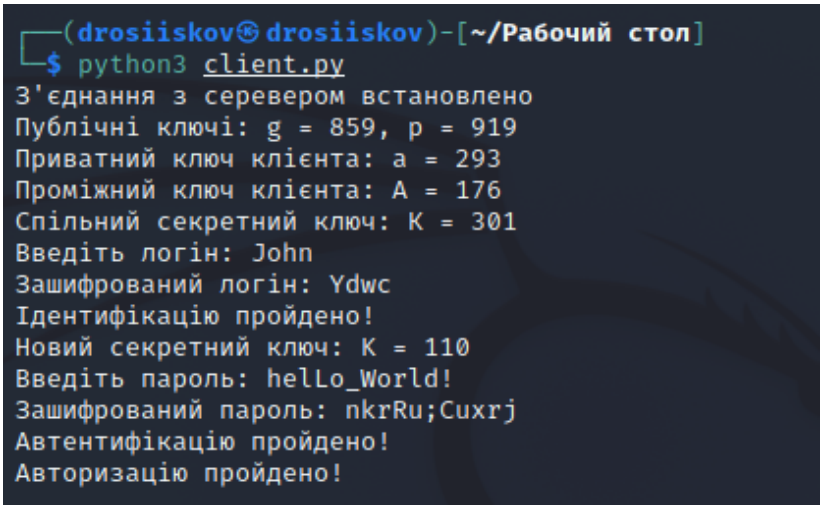


Рисунок 3.21 Робота клієнта на Kali Linux

- Kali Linux – сервер; Windows 10 – клієнт (рис. 3.22 – 3.23).

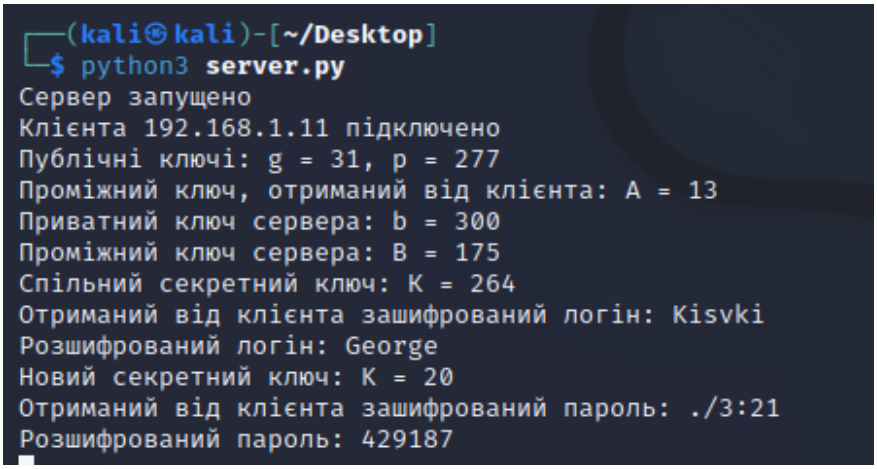


Рисунок 3.22 Робота сервера на Kali Linux

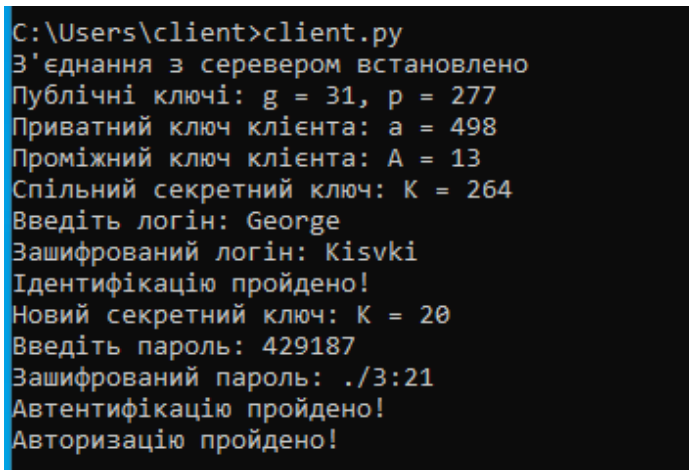


Рисунок 3.23 Робота клієнта на Windows 10

На проходження ідентифікації та автентифікації клієнту виділено по три спроби. Якщо клієнт, вичерпавши спроби, не проходить ідентифікацію (автентифікацію) на сервері, то він отримує відповідне повідомлення. Після вичерпання спроб з’єднання клієнта з сервером припиняється.

Процес проходження ідентифікації клієнта на сервері з вичерпанням усіх спроб наведено на рисунках 3.24 – 3.25.

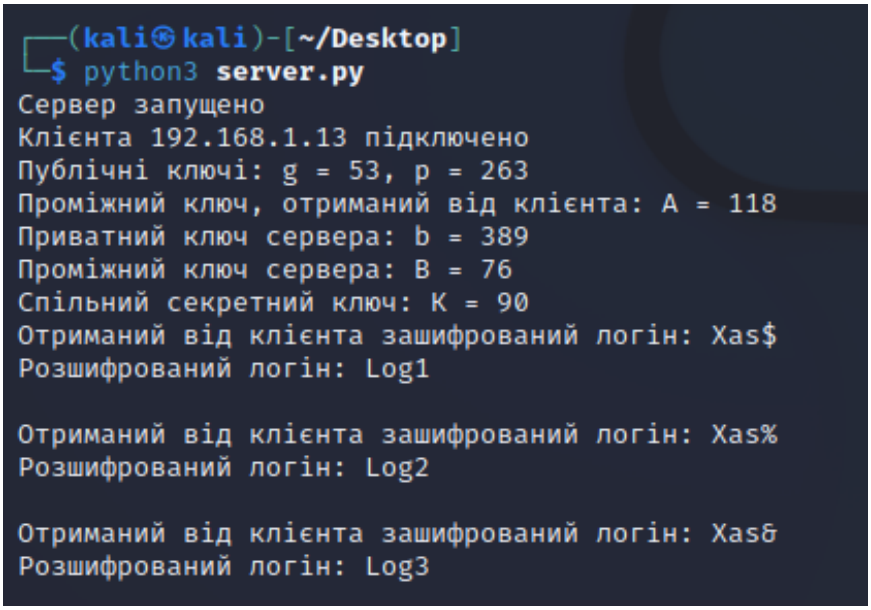


Рисунок 3.24 Невдале проходження ідентифікації клієнта на сервері

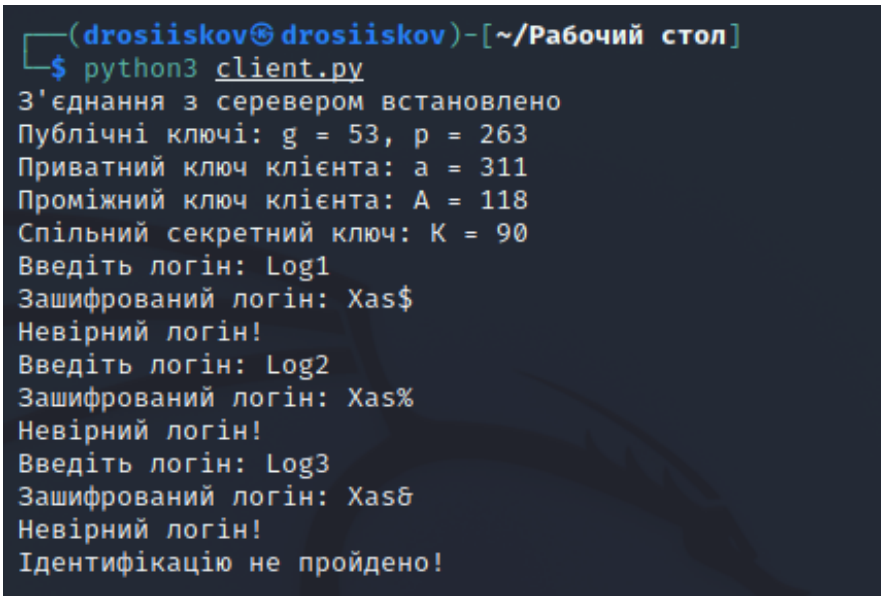


Рисунок 3.25 Результат вичерпання спроб ідентифікації клієнтом

Процес проходження автентифікації можливий лише при вдалому проходженні клієнтом ідентифікації на сервері. При невдалій автентифікації клієнт не авторизується на сервері і з'єднання клієнта з сервером припиняється (рис. 3.26 – 3.27).

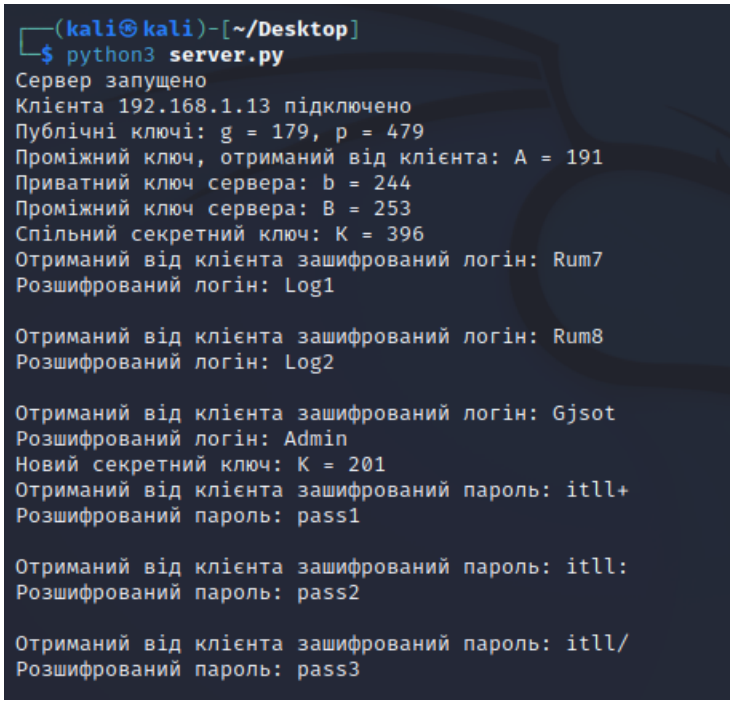


Рисунок 3.26 Невдале проходження автентифікації клієнта на сервері

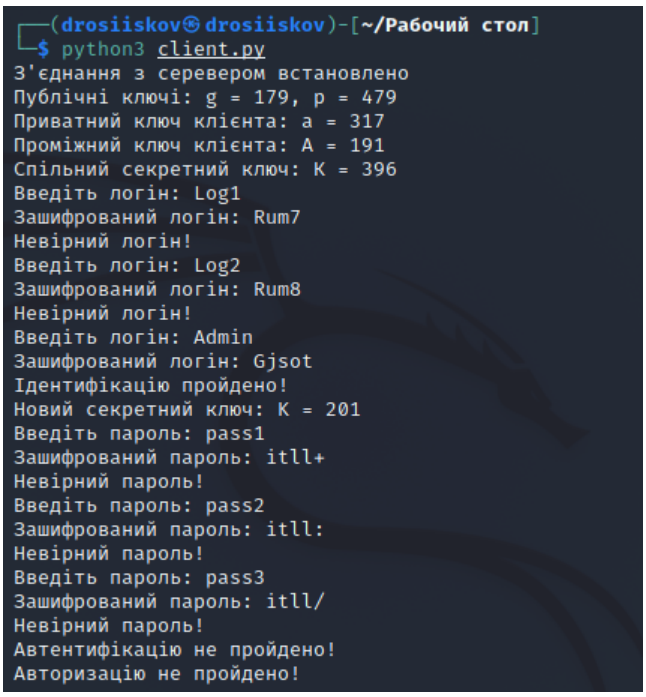


Рисунок 3.27 Результат вичерпання спроб автентифікації клієнтом

Висновки за розділом 3

В даному розділі наведено:

- загальний опис та структуру створеного програмного забезпечення;
- опис використаних модулів, які забезпечують роботу створеного програмного забезпечення;
- опис створених класів, методів та функцій у вигляді тексту та схем, які використовуються в клієнтській та серверній частині програмного забезпечення;
- опис роботи клієнта та сервера та їх взаємодії між собою;
- результати тестування створеного програмного забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.

Код програмного забезпечення наведено в додатку А.

ВИСНОВКИ

В ході виконання роботи:

- Досліджено криптографічні алгоритми шифрування, а саме симетричні та асиметричні алгоритми шифрування. Розглянуто їх основні переваги та недоліки, пов’язані з особливостями їх використання. Також розглянуто можливості компенсації недоліків симетричних та асиметричних алгоритмів, а саме побудову гібридних систем.
- Розглянуто проблему обміну ключами при використанні симетричних алгоритмів шифрування і можливі шляхи її вирішення, такі як використання спеціальних криптографічних протоколів, а саме криптографічного протоколу Діффі-Геллмана.
- Побудовано архітектуру програмного забезпечення, яка є своєрідним шаблоном, за яким створено програмне забезпечення. Визначено основні компоненти, які є необхідними для його створення та тестування, такі як моделі алгоритмів (шифру Цезаря та криптографічного протоколу Діффі-Геллмана), редактор коду, мова програмування, середовище для тестування програмного забезпечення.
- Розроблено програмне забезпечення для обміну секретними ключами на базі протоколу Діффі-Геллмана.
- Описано загальну структуру створеного програмного забезпечення, використані модулі, створені класи, методи та функції, роботу сервера та клієнта з використанням схем.
- Протестовано роботу створеного програмного забезпечення на віртуальних машинах за допомогою системи віртуалізації Oracle VM VirtualBox.

Всі поставлені задачі було виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Порівняння симетричних з асиметричними криптосистемами [Електронний ресурс]: – Режим доступу: https://sites.google.com/site/sucasnikriptosistemik/home/porivnanna-simetricnih-z-asimetricnimi-kriptosistemami.

2. What Is a VPN? - Virtual Private Network [Електронний ресурс]: – Режим доступу: https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html.

3. What Is A Key Exchange? [Електронний ресурс]: - Режим доступу: https://www.jscape.com/blog/key-exchange.

4. Архітектура клієнт-сервер [Електронний ресурс]: – Режим доступу: http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server.

5. Клієнт-серверна архітектура та ролі серверів [Електронний ресурс]: – Режим доступу: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229.

6. КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА [Електронний ресурс]: - Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/.

7. Основи віртуальних машин та їх потенційні проблеми з втратою даних [Електронний ресурс]: – Режим доступу: https://www.ufsexplorer.com/uk/articles/storage-technologies/virtual-machines-data-organization.php.

8. What is the difference between Symmetric and Asymmetric Encryption? Which is better for data security? [Електронний ресурс]: - Режим доступу: https://www.encryptionconsulting.com/education-center/symmetric-vs-asymmetric-encryption/#asym-vs-sym.

9. Symmetric vs. Asymmetric Encryption – What are differences? [Електронний ресурс]: - Режим доступу: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences.

10. Caesar Cipher in Cryptography [Електронний ресурс]: - Режим доступу: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/.

11. Visual Studio Code [Електронний ресурс]: - Режим доступу: https://code.visualstudio.com/.

12. Socket — Low-level networking interface [Електронний ресурс]: - Режим доступу: https://docs.python.org/3/library/socket.html.

13. Advantages and disadvantages of VirtualBox over VMWare [Електронний ресурс]: - Режим доступу: https://www.geekboots.com/story/advantages-and-disadvantages-of-virtualbox-over-vmware.

14. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels [Електронний ресурс] / R. Canetti, H. Krawczyk – Режим доступу до ресурсу: https://iacr.org/archive/eurocrypt2001/20450451.pdf.

15. Socket Programming in Python [Електронний ресурс]: - Режим доступу: https://www.geeksforgeeks.org/socket-programming-python/.

16. Socket Programming in Python (Guide) [Електронний ресурс]: - Режим доступу: https://realpython.com/python-sockets/.

17. Python Advantages and Disadvantages – Step in the right direction [Електронний ресурс]: - Режим доступу: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/.

18. Random — Generate pseudo-random numbers [Електронний ресурс]: - Режим доступу: https://docs.python.org/3/library/random.html.

19. Diffie–Hellman (DH) Algorithm [Електронний ресурс]: - Режим доступу: https://www.hypr.com/diffie-hellman-algorithim/#:~:text=The%20Diffie%E2%80%93Hellman%20(DH)%20Algorithm%20is%20a%20key%2D,or%20data%20using%20symmetric%20cryptography.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного забезпечення

Файл server.py:

```
from random import randint
import socket
import functions
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("192.168.1.74", 4444))
server.listen()
print("Сервер запущено")
users = {"Admin": "sup3rPass_w0rd", "Michael": "qwerty", "John": "helLo_World!", "George": "429187"}
while True:
    user_socket, address = server.accept()
    print(f"Клієнта {address[0]} підключено")
    g, p = functions.Diffie_Hellman.public_keys()
    g_p = str(g) + "," + str(p)
    print(f"Публічні ключі: g = {g}, p = {p}")
    user_socket.send(g_p.encode("utf-8"))
    t_key_A = int(user_socket.recv(2048).decode("utf-8"))
    print(f"Проміжний ключ, отриманий від клієнта: A = {t_key_A}")
    server_prk = randint(11, 1001)
    obj_B = functions.Diffie_Hellman(g, p, server_prk)
    t_key_B = obj_B.first()
    while t_key_B == t_key_A:
        obj_B.private = randint(11, 1001)
        t_key_B = obj_B.first()
```

```
print(f"Приватний ключ сервера: b = {obj_B.private}")
print(f"Проміжний ключ сервера: B = {t_key_B}")
t_key_B = str(t_key_B)
user_socket.send(t_key_B.encode("utf-8"))
result_key = obj_B.res_key(t_key_A)
print(f"Спільний секретний ключ: K = {result_key}")

count_id = 3
count_auth = 3
identification = False
while count_id:
    enc_login = user_socket.recv(2048).decode("utf-8")
    print(f"Отриманий від клієнта зашифрований логін: {enc_login}")
    login = functions.Caesar.decrypt(enc_login, result_key)
    print(f"Розшифрований логін: {login}")
    if functions.identification(users, login):
        user_socket.send("OK".encode("utf-8"))
        identification = True
        break
    else:
        print()
        user_socket.send("Error".encode("utf-8"))
        count_id -= 1
if identification:
    result_key = functions.Diffie_Hellman.exchange_server(user_socket)
    print(f"Новий секретний ключ: K = {result_key}")
    while count_auth:
        enc_pass = user_socket.recv(2048).decode("utf-8")
        print(f"Отриманий від клієнта зашифрований пароль: {enc_pass}")
        password = functions.Caesar.decrypt(enc_pass, result_key)
```

```
            print(f"Розшифрований пароль: {password}")
            if functions.authentication(users, login, password):
                user_socket.send("OK".encode("utf-8"))
                break
            else:
                user_socket.send("Error".encode("utf-8"))
                count_auth -= 1
                print()
    user_socket.shutdown(socket.SHUT_RDWR)
    user_socket.close()
```

Файл client.py:

```
from random import randint, random
import socket
import functions
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("192.168.1.74", 4444))
while True:
    print("З'єднання з серевером встановлено")
    g, p = client.recv(2048).decode("utf-8").split(",")
    g, p = int(g), int(p)
    print(f"Публічні ключі: g = {g}, p = {p}")
    client_prk = randint(11, 1001)
    print(f"Приватний ключ клієнта: a = {client_prk}")
    obj_A = functions.Diffie_Hellman(g, p, client_prk)
    t_key_A = str(obj_A.first())
    print(f"Проміжний ключ клієнта: A = {t_key_A}")
    client.send(t_key_A.encode("utf-8"))
    t_key_B = int(client.recv(2048).decode("utf-8"))
    result_key = obj_A.res_key(t_key_B)
```

```
print(f"Спільний секретний ключ: K = {result_key}")
count_id = 3
count_auth = 3
identification = False
while count_id:
    login = input("Введіть логін: ")
    enc_login = functions.Caesar.encypt(login, result_key)
    print(f"Зашифрований логін: {enc_login}")
    client.send(enc_login.encode("utf-8"))
    answer = client.recv(2048).decode("utf-8")
    if answer == "OK":
        identification = True
        print("Ідентифікацію пройдено!")
        break
    else:
        print("Невірний логін!")
        count_id -= 1
        if count_id == 0:
            print("Ідентифікацію не пройдено!")
if identification:
    result_key = functions.Diffie_Hellman.exchange_client(client)
    print(f"Новий секретний ключ: K = {result_key}")
    while count_auth:
        password = input("Введіть пароль: ")
        enc_pass = functions.Caesar.encypt(password, result_key)
        print(f"Зашифрований пароль: {enc_pass}")
        client.send(enc_pass.encode("utf-8"))
        answer = client.recv(2048).decode("utf-8")
        if answer == "OK":
            print("Автентифікацію пройдено!")
```

```
            print("Авторизацію пройдено!")
            break
        else:
            print("Невірний пароль!")
            count_auth -= 1
            if count_auth == 0:
                print("Автентифікацію не пройдено!")
                print("Авторизацію не пройдено!")
                break
    client.shutdown(socket.SHUT_RDWR)
    client.close()
    break
```

Файл functions.py:

```
import random
class Diffie_Hellman:

    def __init__(self, public_key_1, public_key_2, private_key):
        self.g = public_key_1
        self.p = public_key_2
        self.private = private_key

    def first(self):
        t_key = self.g ** self.private % self.p
        return t_key

    def res_key(self, t_key):
        res_key = t_key ** self.private % self.p
        return res_key
```

```
def public_keys():
    sn = []
    for i in range(11, 1001):
        count = True
        for j in range(2, i):
            if i % j == 0:
                count = False
                break
        if count:
            sn.append(i)
    g = random.choice(sn)
    p = random.choice(sn[10:])
    while g >= p:
        g = random.choice(sn)
    return g, p

def exchange_client(client):
        g, p = client.recv(2048).decode("utf-8").split(",")
        g, p = int(g), int(p)
        client_prk = random.randint(11, 1001)
        obj_A = Diffie_Hellman(g, p, client_prk)
        t_key_A = str(obj_A.first())
        client.send(t_key_A.encode("utf-8"))
        t_key_B = int(client.recv(2048).decode("utf-8"))
        result_key = obj_A.res_key(t_key_B)
        return result_key

def exchange_server(user_socket):
        g, p = Diffie_Hellman.public_keys()
        g_p = str(g) + "," + str(p)
```

```
            user_socket.send(g_p.encode("utf-8"))
            t_key_A = int(user_socket.recv(2048).decode("utf-8"))
            server_prk = random.randint(11, 1001)

            obj_B = Diffie_Hellman(g, p, server_prk)
            t_key_B = obj_B.first()
            while t_key_B == t_key_A:
                obj_B.private = random.randint(11, 1001)
                t_key_B = obj_B.first()
            t_key_B = str(t_key_B)
            user_socket.send(t_key_B.encode("utf-8"))
            result_key = obj_B.res_key(t_key_A)
            return result_key

class Caesar:
    al_up = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    al_low = "abcdefghijklmnopqrstuvwxyz"
    symbols = "!?#$%& ()_+:/,.;0123456789"

    def encypt(text, key):
        enc_text = ""
        for text_let in text:
            if Caesar.al_up.find(text_let) != -1:
                enc_text += Caesar.al_up[(Caesar.al_up.find(text_let) + key) % 26]
            elif Caesar.al_low.find(text_let) != -1:
                enc_text += Caesar.al_low[(Caesar.al_low.find(text_let) + key) % 26]
            elif Caesar.symbols.find(text_let) != -1:
                enc_text += Caesar.symbols[(Caesar.symbols.find(text_let) + key) % 26]
            else:
                enc_text += text_let
```

```
        return enc_text

    def decrypt(enc_text, key):
        dec_text = ""
        key = 26 - key % 26
        for enc_text_let in enc_text:
            if Caesar.al_up.find(enc_text_let) != -1:
                dec_text += Caesar.al_up[(Caesar.al_up.find(enc_text_let) + key) % 26]
            elif Caesar.al_low.find(enc_text_let) != -1:
                dec_text += Caesar.al_low[(Caesar.al_low.find(enc_text_let) + key) %
26]
            elif Caesar.symbols.find(enc_text_let) != -1:
                dec_text += Caesar.symbols[(Caesar.symbols.find(enc_text_let) + key) %
26]
            else:
                dec_text += enc_text_let
        return dec_text
def identification(users, login):
    identification = False
    for i in users:
        if login == i:
            identification = True
    if identification:
        return True
    else:
        return False
def authentication(users, login, password):
    if password == users.get(login):
        return True
    else:
```

```
return False
```

ДОДАТОК Б
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИПЛОМУ

Тези наукових доповідей

1. Даков С.Ю., Російськов Д.С., Використання криптографічних протоколів для обміну секретними ключами в клієнт-серверній архітектурі, IV Міжнародна науково-практична конференція "Проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS)" 15 – 16 КВІТНЯ 2021, КИЇВ, Україна.

2. Даков С.Ю., Російськов Д.С., Використання VPN технологій для створення захищених каналів зв’язку, V Міжнародна науково-практична конференція "Проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS)" 2022, КИЇВ, Україна.